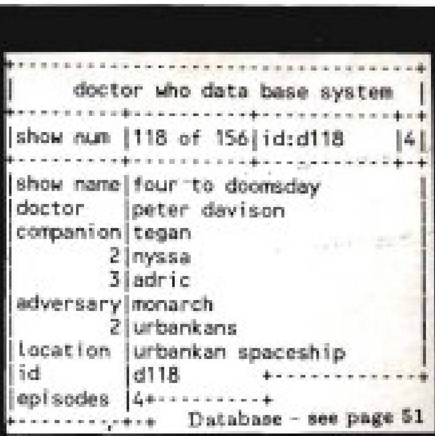
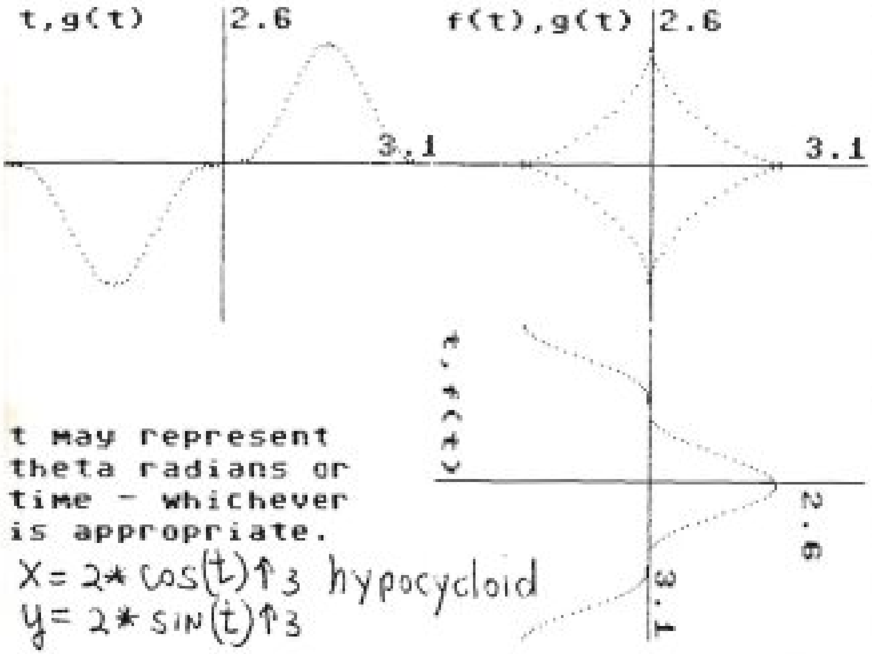
COMAL TODAY 23



```
+-----------------------------------------------+
|      doctor who data base system              |
+-----------+------------+----------------+-----+
|show num   |118 of 156|id:d118           |4|
+-----------+------------+----------------+-----+
|show name|four to doomsday
|doctor    |peter davison
|companion|tegan
|         2|nyssa
|         3|adric
|adversary|monarch
|         2|urbankans
|location |urbankan spaceship
|id        |d118         +-------------+
|episodes |4+----------+
+---------+--+   Database - see page 51
```



t may represent theta radians or time - whichever is appropriate.

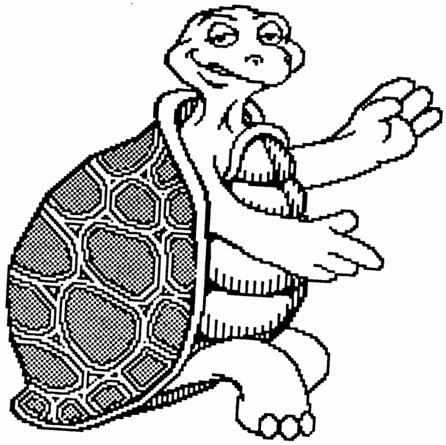
X= 2*cos(t)↑3 hypocycloid
y= 2*sin(t)↑3

Graphing Parametric Equations - see page 32

COMAL Today
5501 Groveland Ter
Madison, WI 53716

If the label says
Last Issue: 23
You must renew now.
Use order form inside.
No other notice is sent.

COMAL Benchmarks



COMAL is now running on many different computers. Just for fun, we took our PRIME number SIEVE program, and ran it on every COMAL (even the preliminary ones) that we knew about in North America. The program was run twice, once printing the numbers as they were found, and again without printing them. The results shown below are within a second.

Note: these tests show two comparisons. In some cases, the same COMAL is being run on two different computers. CP/M COMAL runs much faster on the Kaypro than on the C128. In the other case, the same computer is used to run two different versions of COMAL. German Amiga COMAL was over four times faster than Mytech Amiga COMAL on an Amiga 500.

<u>NOT PRINTING NUMBERS (in seconds):</u>

1	Tandy 4000 (80386) UniComal **2.2**
5	**German Amiga COMAL 2.0 prelim**
8	UniComal IBM PC COMAL **2.2**
13	C64 COMAL 2.0 on C128 FAST
13	C128 COMAL 2.0 FAST
21	MacIntosh COMAL 2.0 **(cancelled)**
26	Mytech Amiga COMAL 2.0 prelim
28	C64 COMAL 2.0
28	C128 COMAL 2.0
28	PET 8096 COMAL 2.0 (ROM board)
31	CP/M COMAL 2.10 on Kaypro
35	CP/M COMAL 2.10 on Epson
38	Mytech IBM PC COMAL **(cancelled)**
50	**Amiga Basic (with si#=3000)**
65	Apple COMAL 1.0 prelim
67	C64 **Power Driver** / COMAL 0.14
72	PET 8032 COMAL 0.14
87	CP/M COMAL 2.10 on C128

<u>PRINTING NUMBERS (in seconds):</u>

2	Tandy 4000 (80386) UniComal **2.2**
13	UniComal IBM PC COMAL **2.2**
17	**German Amiga COMAL 2.0 prelim**
21	C128 COMAL 2.0 FAST
28	C64 COMAL 2.0 on C128 FAST
38	CP/M COMAL 2.10 on Kaypro
39	PET 8096 COMAL 2.0 (ROM board)
40	C64 COMAL 2.0
43	C128 COMAL 2.0
45	CP/M COMAL 2.10 on Epson
54	Mytech IBM PC COMAL 2.0 **(cancelled)**
74	Mytech Amiga COMAL 2.0 prelim
76	Apple COMAL 1.0 prelim
77	MacIntosh COMAL 2.0 **(cancelled)**
81	C64 **Power Driver** / COMAL 0.14
84	PET 8032 COMAL 0.14
111	CP/M COMAL 2.10 on C128

<u>Here is the program we used</u>:

```
si#:=3962; count#:=0
DIM flags#(0:si#)
FOR i#:=0 TO si# DO
  IF NOT flags#(i#) THEN
    prime#:=i#+i#+3
    count#:+1
    //print prime#;
    FOR k#:=i#+prime# TO si# STEP prime# DO
      flags#(k#):=TRUE
    ENDFOR k#
  ENDIF
ENDFOR i#
PRINT "count=";count#
PRINT "last prime =";prime#
```

<u>NOTES</u>

- Mytech COMAL doesn't initialize elements in an array automatically as it should. Thus, it needed an extra line to fill the array with 0.
- A Zenith 151 is our IBM PC compatible.
- PET COMAL 0.14 & **Amiga Basic** didn't have enough room to run the full array size. We used a smaller size and estimated what the result would have been with the full array.
- **Changes / additions since issue #19 are bold.**
- Thanks to Jeffery Ziebelman at Madison's Radio Shack Computer Center for allowing us to run the program on their new Tandy 4000 computer system.
- **Thanks to Richard Barton for the use of his Amiga 500.**

COMAL Today Issue 23 October 29, 1988

General

Fun & Graphics

2.0 Packages & Programming

Applications

Reference

Editor
Len Lindsay

Assistant
Maria Lindsay

Contributors
Lewis Brown
Jim Frogge
German Amiga Co.
Dawn Hux
Luther Hux
Bill Inhelder
Paul Keck
Len Lindsay
Ed Matthews

Contributors
Gary Parkin
Joel Rea
Carmen Sorvillo
David Stidolph
James Synnamon
UniComal
David Warman
Robert Webb

COMAL Today is published by COMAL Users Group, U.S.A., Limited, 5501 Groveland Ter, Madison, WI 53716 and welcomes contributions of articles, manuscripts and programs which would be of interest to readers. All manuscripts and articles sent to COMAL Today will be treated as unconditionally assigned for publication and copyright purposes to COMAL Users Group, U.S.A., Limited and is subject to the Editor's unrestricted right to edit and to comment editorially. Programs developed and submitted by authors remain their property, with the exception that COMAL Users Group, U.S.A., Limited reserves the right to reprint the materials, based on that published in COMAL Today, in future publications. There will be no remuneration for any contributed manuscripts, articles or programs. These terms may be varied only upon the prior written agreement of the Editor and COMAL Users Group, U.S.A., Limited. Interested authors should contact the Editor for further information. All articles and programs should be sent to COMAL Users Group, U.S.A., Limited, 5501 Groveland Ter, Madison, WI 53716. Authors of articles, manuscripts and programs warrant that all materials submitted are original materials with full ownership rights resident in said authors. The opinions expressed in contributed articles are not necessarily those of COMAL Users Group, U.S.A., Limited. Although accuracy is a major objective, COMAL Users Group, U.S.A., Limited cannot assume liability for article or program errors.

Our NEW Address is:

COMAL Users Group, U.S.A., Ltd.
5501 Groveland Terrace
Madison, WI 53716
(608) 222-4432

Editor's Disk

by Len Lindsay

Make sure you have changed our address in your address book (or computer). Our Monona Drive address is no longer valid. Mail sent to it tends to get lost. Unfortunately, the old address still is in the boot files of many disks and on some of our books. So, no matter what the disk or book says, the current correct address is:

COMAL Users Group, U.S.A, Ltd
5501 Groveland Terrace
Madison, WI 53716

It took longer than usual, but here is issue 23. Due to a lack of income, there is not enough money to pay for a staff. Even Maria and I have not been paid a salary since January. Luckily, with Amiga and Apple COMALs out, I hope our situation improves.

Meanwhile I still have my regular full time job running an IBM mainframe computer for the State of Wisconsin. Plus two part time jobs! In addition to that, I type in orders here and take care of all the *business* work required. Maria does the packing and shipping of the orders. After that comes the newsletter. I do it on my own in the very little time left.

Needless to say, I have no time left to work on or help with new COMAL books. My apologies to the authors. Perhaps someone else out there can co-ordinate and set up masters for new books. Several are in the works and have been horribly delayed already.

Amiga users take note. There now are not one, but two COMALs for the Amiga. This issue David Stidolph gives a quick impression of his favorite, the one from Germany. We had hoped to have an article about the Mytech Amiga COMAL for this issue, but it did not arrive. We now hope it will be in the next issue. Both implementations are affordable (under $100). German Amiga COMAL went all out to be as UniComal like as possible. I like that. By the time you read this I should have ordering information for it. Give us a call, or send me a Self Addressed Stamped Envelope for details.

In my long article on programming details, I mention IBM and C64 COMAL. Please note that the CP/M COMAL is very similar in capability to IBM COMAL. I started to add it into the article, but it made the article hard to read, so I left it out. The German Amiga COMAL should also be just like the IBM one (both C64 and IBM COMALs are by UniComal).

Perhaps you have heard about the law that congress may pass requiring Mail Order companies to collect a state sales tax for every state. Needless to say, I have a hard time keeping records for just Wisconsin. I would not have time to modify my order system to keep track of over 40 different state sales tax percentages and running totals. Nor would I have time to fill out over 40 sales tax forms per year. If the government ever requires this of COMAL Users Group, U.S.A., Limited, it would be the same as asking us to shut down. I hope they don't pass the law, but at least now you are aware of how it would affect small companies. It is not the money for the sales tax that is significant. It is the extra work.

I try to maintain my sanity by taking a break from reality each week ... you guessed it. I watch Doctor Who on our local PBS station. If you need a break, check out the inside back cover for a partial list of TV stations that show Doctor Who. The list was compiled with the help of QLink Who fanatics. Please send me a postcard with your local PBS station and the day / time it shows Doctor Who.

Back to COMAL. I think this issue has a good variety of material. Lots of listings! I already have a start on the next issue. If you get the new Amiga COMAL, please consider sending in an article about it ... as well as programs. ■

Bugs

BUG notice posted on QLink

SUBJ: Bug in DIR DESIGNER? FROM: Xojo
After I put some separating lines in a directory (using dir'designer), then tried to validate it, I would get a message something like:

00, invalid track or sector, 75 01

The program fools the DOS into thinking that the "prettifying" directory entries are USR files of length 0 that own track 18 sector 18 (the last block in a full 144 entry directory). But DOS expects to see the first bytes of ANY last block contain 0 & 255 (or zero followed by the actual number of bytes used in the final block; if the directory uses it, it would be 0 & 255). Any disk directory with 137 or more entries (now or any time in the past) is OK. Once the last possible block (18 18) is marked it stays marked. My fix: write 0 & 255 into the first bytes of trk 18 sctr 18 everytime the directory is rewritten. Add these lines to PROC write'dir:

```
4265   CLOSE FILE 2
4266   string$:=""0""255"" // my fix
4267   write'block(18,18)  // my fix
4268   CLOSE FILE 2        // my fix
4270 ENDPROC write'dir
```

SUBJ: Dir Designer FROM: DavidW57
I am the author of the program on Today Disk #22. When I was testing that part of the program, the link bytes in T18, S18 had already been set, which explains why I never got the error. It sounds like your fix should work. I should be at the Thursday QLink COMAL meeting if you want to discuss it there. Here are some changes. In PROC write'dir:

```
4265 CLOSE FILE 2 //  «---delete this line
4266 string$:=""0"" // add this line
4267 write'block(18,18) // add this line
```

Another bug occurs if you do something else before dropping an entry you pick up. Change two procs to fix it. In PROC perform:

```
1572 temp'valid$:=valid$ // add this line
1573 valid$:=""13""145""17"" // add this line
1582 valid$:=temp'valid$ // add this line
```

In PROC copy'entry:

```
3132 temp'valid$:=valid$  // add this line
3133 valid$:=""13""145""17"" // add this line
```

Also, when copying an entry, I forgot to update the total file count. So, in PROC place'entry:

```
2402 file'count // add this line
```

Delete the debugging aids I left in the program. All lines containing the variable no'error can be deleted (there are 3 of them -- use FIND). And in PROC write'block:

```
3950 STOP // «---delete this line
```

¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤

Reorder Directory BUG

In Reorder Directory, *COMAL Today #22*, page 32, the directory is saved correctly to disk, but, the screen / printer section of the program won't always list the correct track and sector a directory block will be saved to. To fix this:

```
FIND "dir'sec((" //locates 2 lines to change
```

In the first line found add an extra set of () and change +3 to be +1 to get:

```
IF i<=ne THEN PRINT FILE 7: TAB(35),"18-
",STR$(dir'sec(((i-1) DIV 8)+1)),
```

Make corresponding changes to the 2nd line:

```
IF q<=ne THEN PRINT TAB(35),"18-",STR$(
dir'sec(((q-1) DIV 8)+1)),
```

Notice that the second line has **q** in place of **i**, and PRINT in place of PRINT FILE 7. ■

Amiga COMAL

by David Stidolph

When it rains, it pours. And it is definitely pouring COMAL this month. Not one, not two, but three new COMAL systems are on the verge of being released. I played with all three. For now I will only talk about the one I am most impressed with. No, not Apple II COMAL, the system I am writing (though it should be released by the time you read this). No, not Mytech Amiga COMAL.

The best news is from Germany. A small team of developers have been secretly hard at work, with direct help from the founder of COMAL himself, Borge Christensen! One of the developers was the main programmer behind CP/M COMAL 2.1. We knew he was an excellent programmer, when he created a modern COMAL to run on the ancient operating system. What now? Is he even better, creating a futuristic COMAL for a modern computer system? Find out for yourself! This hot new COMAL will soon be distributed in the USA... probably by the COMAL Users Group, USA, Ltd... and probably for just $99.

[*To make this article easier to understand, I will refer to the new Amiga COMAL from Germany as* *<u>Amiga COMAL</u>*.]

Just days before this issue was to go to press, a preliminary copy of Amiga COMAL came in the mail. Wonderful, except that I don't have an Amiga (yet). So, I rushed over to a friends house to use his Amiga. These are my first impressions. (*Special thanks to Richard Barton for the use of his Amiga 500*). Please keep in mind that I am writing this for those who are somewhat familiar with the Amiga.

Amiga COMAL starts with two windows active. One for program execution output; the other for entering programs and commands. However, if you wish you can set program output to go to the command screen with the command:

```
runwindow-
```

Now your screen is just like your favorite C64 screen. Amiga COMAL provides a full screen editor. It works like the C64 editor! If you see a mistake on a line, just move the cursor to that line, correct the error, and press «*return*».

Amiga COMAL goes even further. It works with a virtual window rather than the actual video window. The advantage to this is apparent when you resize the window. The text that was in the old window is re-drawn immediately. Yes, this is a pleasant surprise to an Amiga user.

The INSTALL program sets the free memory size. It is limited only by the amount of memory in your computer (by default it is 64K free). Imagine having a one or two megabyte program - just think of the editing possibilities.

Packages

The C64 COMAL 2.0 cartridge brought a revolution in programming with it ... packages! Cartridge programmers could enhance the language with machine code routines. A program uses those routines just like built in COMAL procedures and functions.

Amiga COMAL not only allows you to write packages in assembly code, but also in C, FORTH, Modula II or any other language that produces machine code. If you are like most new programers, however, the task of writing a C program is about as exciting as a trip to the dentist. If only there was an easier way.

Now there is. With Amiga COMAL you can write packages in COMAL. Yes, COMAL. Just write your package as a normal COMAL program -- even call other packages -- and save it to disk with the extension **.pck**. That's all there is to it... you just wrote a package!

more»

A package is divided into three parts:

1. **Initialization**
2. **Procedures and Functions**
3. **Signal routine**

The following is an example package:

```
0010 epsilon:=0.0000001
0020 //
0030 FUNC test'close(num1, num2) CLOSED
0040   num3:=(ABS((num1-num2)/num2)
0050   RETURN num3<=epsilon
0060 ENDFUNC test'close
0070 //
0080 PROC signal(s) CLOSED
0090   IF s=5 THEN // 5 means RUN
0100     epsilon:=0.0000001
0110   ENDIF
0120 ENDPROC signal
```

In the example, the **initialization** section is only one line (0010). It can be any length. It is executed only once, at load time. The **signal** procedure is called whenever the following situations occur:

1 = USE
2 = DISCARD
3 = NEW
4 = unused
5 = RUN
6 = CON
7 = Program ERROR
8 = END
9 = Normal end/STOP/ESC
10= BYE

In the example, the **signal** routine checks if the RUN command has been issued (a parameter of 5 means a RUN command is to be executed). Each time RUN is issued, our example package resets the value of **epsilon** to 0.0000001. We created a global variable. **Epsilon** is available to any other program. After you save the program to disk, you can use it as a package:

```
  SAVE "epsilon.pck"
```

To use the package, include a USE command in the program. For example:

```
0010 USE epsilon
0020 //
0030 INPUT "Enter 1st number: ": n1
0040 INPUT "Enter 2nd number: ": n2
0050 CASE test'close(n1, n2) OF
0060 WHEN TRUE
0070   PRINT "Numbers are equal"
0080 WHEN FALSE
0090   PRINT "Numbers not equal"
0100 ENDCASE
```

When the program is RUN all USE statements are scanned and the packages brought in from disk (using the name given in the USE statement plus **.pck** as the filename). We use **test'close** in our example program as if it were a built in function. We defined it in our example package. It tests if two numbers are close enough to being equal that the difference may be just computer round off error.

Prior to Amiga COMAL, only advanced programmers had the luxury of creating packages. Now, even beginners can try their hand at it. You can make a simple program into a package just by saving it to disk with the extension **.pck**. However, the real fun begins when you take advantage of signal. The possibilities are endless. Here are some ideas:

1 - USE

Every time a USE command is about to be executed, your package can do something first. You may wish to prompt the user to insert a special disk, or initialize an array.

2 - DISCARD

Your package is told - in advance - when it is about to be discarded. This allows you to put things back the way they were if your package messes with the system.

3 - NEW

Each time a NEW command is about to be executed, your package will have the chance to

more»

do something ... even just print a message on the screen (like "*Goodbye program.*")

5 - RUN
Every time a program is RUN your package can do something first, like reset the screen colors.

6 - CON
Each time a program is continued after stopping, your package can first reset the screen colors, for example.

7 - program ERROR
This gives your package the capability to be a giant error trap! You can give COMAL GURU messages!

8 - END & 9 - program end/STOP/ESC
As a program ends, your package has the ability to do something, even erase the program! Talk about unlistable programs!

10 - BYE
Your package even gets the last word in before COMAL itself shuts down!

Passing Procedures As A Parameter

One interesting new feature in Amiga COMAL is the ability to pass a procedure or function to another procedure/function as a parameter. This is something I don't expect many to use, but here is an example:

```
0010 PROC do'proc(REF p)
0020  EXEC p
0030 ENDPROC do'proc
0040 //
0050 PROC a
0060   PRINT "Inside procedure a"
0070 ENDPROC a
0080 //
0090 PROC b
0100   PRINT "Here I am, Procedure b"
0110 ENDPROC b
0120 //
0130 do'proc(a)
0140 do'proc(b)
```

```
RUN
Inside procedure a
Here I am, Procedure b
```

Notice the two lines printed by the program. The interesting thing is that Richard Bain and I spent hours discussing how this would be impossible. I now join those who said we'd never reach the moon. Amiga COMAL did it!

PASSing Commands to CLI

Just like the PASS command in UniComal's IBM version, you can send commands to the operating system. This is a VERY powerful feature. It means that you can perform ANY operating system feature from COMAL.

TRACE Program Execution

Like CP/M COMAL you can trace the execution of a program, or even single step it. An entire article could be devoted to just some of the advantages of TRACE. Perhaps someone can write that article for a future issue.

Last Impressions

After just a short time with Amiga COMAL, I found it the easiest to use, and the closest to the UniComal 2.0 standard. The language is fast (five times faster than Mytech on a simple numeric benchmark), and very powerful. I **recommend it to all who want to program on the Amiga.**

I planned to tell you about the problems and bugs I encountered testing the preliminary release. Unfortunately, I found no problems in the language, just a couple corrections needed in the command level system (*I think I caused them myself by not running* INSTALL). [Note: *preliminary specifications are subject to change*]

UniComal New Products

We have several announcements from UniComal. IBM PC COMAL 2.2 will be the current version through at least March of 1989, when they may announce a new version at the Hannover Fair in Germany where they have booked a stand.

IBM PC COMAL 2.2

This is the current version of COMAL for the IBM PC or PS/2. It is the fastest COMAL out and the one I use for my work. It comes packaged in a UniComal Doc Box with a huge reference manual, spiral bound tutorial book, quick reference guide, and 3 disks (system disk, tutorial disk, and supplemental programs disk). It includes Graphics and Sound packages, and supports the 80x87 math co-processor. Special order price is $495 plus $5 shipping/handling. There is a $50 discount if an order is prepaid.

IBM PC COMAL 2.2 PLUS

All of the above plus a compiler and serial communications package (SCOM). This adds one more binder, another reference manual, and two more disks. I use this compiler to distribute programs I write in COMAL for the IBM. A compiled program is a stand-alone file, and can be distributed without royalties. Special order price is $795 plus $7 shipping/handling. There is a $100 discount if the order is prepaid.

Upgrades

You now may upgrade 2.2 to 2.2 PLUS (with the added compiler). The cost is $300 and requires your UniComal registration number for the 2.2 version you own.

You can upgrade 2.1 to 2.2. The cost is $45 and requires your 2.1 UniComal registration number.

You can upgrade 2.1 PLUS to 2.2 PLUS. The cost is $45 and requires your 2.1 PLUS UniComal registration number.

The Quick Reference Booklet now comes with the 2.2 systems, but is available separately to previous purchasers. Special order price: $20.

Page Dividers for ref manual are now included with the 2.2 system. These nice heavy duty custom printed page dividers are now available separately for $8.

School License

After purchasing one regular IBM PC COMAL, schools can get additional copies with this license. Each additional copy without documentation is $70. Each addition set of documentation is $55.

UniDump

This makes it possible to dump a graphics screen on a laser printer, HP Thinkjet, NEC P6/P7 or other printers. It is activated by pressing the SHIFT PrtSc key, or by calling the printscreen procedure. It replaces the printscreen procedure in the graphics package. Printing is done in portrait mode (not rotated 90 degrees). Seven different ways of printing is available, if the printer supports the modes. Special order price is $45.

UniMatrix

The UniMatrix package performs matrix operations rapidly and efficiently. It includes procedures and functions to perform the following types of matrix calculations:

- Fundamental matrix operations, such as rounding all elements, computing the absolute values of all elements and various types of addition, subtraction, multiplication and division.
- Linear simultaneous equations can be solved; determinants, condition numbers and other useful matrix quantities can be calculated.

more»

- It is possible to manipulate matrix elements in various ways. For example diagonal matrices and identity matrices can be defined, or selected submatrices can be inserted or removed from a given matrix.

- Specified elements and their row and column positions within a given matrix can be found. For example, maximum or minimum values can be found, and the elements of the matrix can be added together.

- Facilities are provided for transforming vectors to matrices and vice-versa. Both the scalar and vector products of vectors are defined.

UniMatrix requires an 80x87 co-processor installed in the computer. Full use is made of available memory. DOS memory outside the UniComal data/program area is used for intermediate matrix calculations if necessary.

Intermediate calculations are carried out to 18-19 digits precision. The 80x87 co-processor operates in parallel to the 80x86 / 8088 processor to achieve optimum speed. Special order price is $165.

Hercules Graphics Support

This makes it possible to run graphics on a monochrome Hercules monitor. Special order price is $85.

Btrieve Interface

This allows you to "hook" into Btrieve with UniComal IBM PC COMAL 2.2. Btrieve is Novell's key-indexed file management system that can be used with any programming language (including UniComal) for high performance file handling and improved programming productivity. Btrieve's fault tolerant processing guarantees data integrity without additional programming. Based on the b-tree indexing method and implemented with cache buffers, Btrieve delivers fast, maintenance free operation. Btrieve provides maximum speed in accessing data. It requires MS-DOS 3.x, OS/2, or PC MOS/386. Estimated price is $245. Special order for UniComal's interface to Btrieve is $25 single user, $110 multi-user.

[*Special note: from my preliminary research, Btrieve sounds spectacular. It sounds very reliable as well as lightning quick. And now you can hook into it from IBM PC COMAL.*]

XQL Interface

This allows you to "hook" into XQL, with its structured query language, directly from UniComal IBM PC COMAL 2.2. XQL is Novell's relational database management system designed for programers using UniComal, COBOL, BASIC, Pascal, or C. XQL allows users to access their databases with the ease of Structured Query Language (SQL). In addition, XQL frees an application from physical file characteristics by providing true relational capabilities with data independence, data descriptions, data integrity, and security. XQL reduces programming time, enhances application capability, and improves application performance. It includes 19 Relational Primitives and about 100 commands. The Primitives Manager requires 97K-187K. The SQL Manager requires 80K-118K plus the Primitives Manager. It requires MS-DOS 3.x or OS/2 and Btrieve 4.11 or later. Estimated price is $795. UniComal's interface to XQL is $110, special order price.

[*Special note: if you can't get Btrieve or XQL locally, we can get it for you, at a discount from the estimated price.*] ■

Message Board

by Ed Matthews

The "*Electronic Message Board*" program was written for displaying messages in an eye catching format in our department at Southwest Missouri State University. We are using the compiled Power Driver version of the program on a C64 with a 1702 monitor, and it has become quite popular.

Messages are displayed in the sequence they appear in the text, and each time the screen is cleared, the text and background (including border) colors change to a random combination; if the combination is not one determined to be legible, the random function repeats until the combination is acceptable.

The message file is simply a sequential text file in PET ASCII, with margins of 1 and 39. Since it is useful to know how old the information is, the first line of the text file should be "Updated (date)," and will be displayed at the top of every screen. Since Commodore doesn't allow certain characters in sequential files, some substitutions are made:

Use percent	(%)	for comma (,)
Use "at"	(@)	for colon (:)
Use asterisk	(*)	for a space at the beginning of a line.
Use plus	(+)	for cursor down.
Use pound	(#)	for clear screen and change colors.
Use equals	(=)	for 1 second pause.

I use PaperClip and its "PET ASCII" printer drive, printing to device 8, the disk drive. Top margin is 0, and paging is at 66, so there are no blank lines in the text. Since Commodore won't allow blank lines in a SEQ file, put an asterisk (*) on a line or use the plus (+) for cursor down.

Being able to set colors under program control might be a worthwhile enhancement. Message categories could always be displayed in particular color combinations. Making the program interactive, so you could have all of a category of messages displayed when you wanted it, would be useful, too.

```
//save "msgboard" // 04-14-88 1145
setup // for C64 2.0
dims
disk'input
run'display
//
PROC setup
  PRINT CHR$(9),CHR$(14),CHR$(8)
  USE system
  USE graphics
ENDPROC setup
//
PROC dims
  DIM character$ OF 1
  DIM file'name$ OF 20
  DIM screen'line$(1:235) OF 39
  speed:=1000
  the'cows'come'home:=FALSE
ENDPROC dims
//
PROC disk'input
  PRINT CHR$(147),CHR$(17)
  PRINT "Retrieving Text File from Disk"
  disk'error
  PRINT "Disk Directory?  (Y or N) ";
  IF inkey$ IN "Yy" THEN
    PRINT
    PRINT "  Enter name or first characters."
    PRINT "  For complete directory, press
    «return»" //wrap line
    PRINT
    INPUT AT 0,6: "": file'name$
    IF LEN(file'name$)>15 THEN file'name$
    :=file'name$(1:15) //wrap line
    file'name$:="0:"+file'name$+"*=s"
    disk'dir(file'name$)
  ENDIF
  PRINT
  PRINT "  What text file will you use?"
  PRINT
```

more»

Message Board - continued

```
  INPUT AT 0,10: "": file'name$
  OPEN FILE 8,file'name$,READ
  line:=1
  REPEAT
    INPUT FILE 8: screen'line$(line)
    FOR element:=1 TO LEN(screen'line$(line))
      CASE screen'line$(line)(element) OF
      WHEN "%"
        screen'line$(line)(element):=","
      WHEN "@"
        screen'line$(line)(element):=":"
      WHEN "#"
        screen'line$(line)(element):=CHR$(147)
      WHEN "+"
        screen'line$(line)(element):=CHR$(17)
      WHEN "*"
        screen'line$(line)(element):=" "
      OTHERWISE
        NULL
      ENDCASE
    ENDFOR element
    PRINT screen'line$(line)
    line:+1
  UNTIL screen'line$(line-1)="$$end"
  total'lines:=line-1
  CLOSE FILE 8
  FOR count:=1 TO speed DO NULL
ENDPROC disk'input
//
PROC disk'dir(file'name$)
  PRINT
  PRINT "  Press space bar to pause."
  DIR file'name$
ENDPROC disk'dir
//
PROC disk'error
  PRINT
  PRINT "   Checking disk drive."
  REPEAT
    PASS "i0"
    error:=FALSE
    IF STATUS$<>"00, ok,00,00" THEN
      error:=TRUE
      PRINT "  This disk drive is not ready"
      PRINT "  Try again?";
      IF NOT inkey$ IN "Yy" THEN END
    ENDIF
  UNTIL error=FALSE
  PRINT
ENDPROC disk'error
//
PROC run'display
  REPEAT
    FOR line:=2 TO total'lines DO
      bracket:=TRUE
      position:=1
      REPEAT
        character$:=screen'line$(line)(position)
        CASE character$ OF
        WHEN "="
          FOR seconds:=1 TO speed DO NULL
        WHEN CHR$(147)
          change'colors
          PRINT CHR$(147)
          PRINT AT 1,(40-LEN(screen'line$(1
          ))):screen'line$(1) //wrap line
          PRINT
        WHEN "]"
          change'backg
        WHEN "["
          change'text
        WHEN "&"
          PRINT CHR$(147)
        OTHERWISE
          PRINT character$,
        ENDCASE
        position:+1
      UNTIL position>LEN(screen'line$(line))
      PRINT
    ENDFOR line
  UNTIL the'cows'come'home
ENDPROC run'display
//
PROC roll'message
  DIM string$ OF 300
  DIM short'string$ OF 63
  short'string$:="Help! My name is Ernie. I'm
  trapped inside this computer"//wrap line
  blank$:=SPC(43)
  FOR count:=1 TO 2 DO
    string$:=string$+short'string$
  ENDFOR count
```

more»

```
  PRINT CHR$(147)
  FOR count:=1 TO 29 DO
    PRINT AT 20,5: blank$(1:30-count
    ),string$(1:count), //wrap line
    FOR wait:=1 TO 20 DO NULL
  ENDFOR count
  FOR count:=1 TO LEN(string$)-30 DO
    PRINT AT 20,5: string$(count:count+30)
    FOR wait:=1 TO 50 DO NULL
  ENDFOR count
  FOR count:=LEN(string$)-29 TO LEN(string$)
    PRINT AT 20,5: string$(count:LEN(string$
    )),"  " //wrap line
    FOR wait:=1 TO 50 DO NULL
  ENDFOR count
  PRINT AT 20,5: "  "
ENDPROC roll'message
//
PROC change'colors
  CASE character$ OF
  WHEN CHR$(147)
    backg:=RND(1,15)
    CASE backg OF
    WHEN 1
      REPEAT
        text:=RND(2,14)
        CASE text OF
        WHEN 2,4,5,6,8,9,14
          combo:=TRUE
        OTHERWISE
          combo:=FALSE
        ENDCASE
      UNTIL combo=TRUE
    WHEN 2
      REPEAT
        text:=RND(1,15)
        CASE text OF
        WHEN 1,7,0,13,15
          combo:=TRUE
        OTHERWISE
          combo:=FALSE
        ENDCASE
      UNTIL combo=TRUE
    WHEN 3
      REPEAT
        text:=RND(0,11)
        CASE text OF
        WHEN 0,2,4,6,8,9,11
          combo:=TRUE
        OTHERWISE
          combo:=FALSE
        ENDCASE
      UNTIL combo=TRUE
    WHEN 4
      REPEAT
        text:=RND(0,15)
        CASE text OF
        WHEN 0,1,3,7,13,15
          text:=0
          combo:=TRUE
        OTHERWISE
          combo:=FALSE
        ENDCASE
      UNTIL combo=TRUE
    WHEN 5
      REPEAT
        text:=RND(0,11)
        CASE text OF
        WHEN 0,1,2,6,9,11
          combo:=TRUE
        OTHERWISE
          combo:=FALSE
        ENDCASE
      UNTIL combo=TRUE
    WHEN 6
      REPEAT
        text:=RND(1,13)
        CASE text OF
        WHEN 1,3,7,13
          combo:=TRUE
        OTHERWISE
          combo:=FALSE
        ENDCASE
      UNTIL combo=TRUE
    WHEN 7
      REPEAT
        text:=RND(0,9)
        CASE text OF
        WHEN 0,4,6,8,9
          combo:=TRUE
        OTHERWISE
          combo:=FALSE
```

more»

```
        ENDCASE
      UNTIL combo=TRUE
    WHEN 8
      REPEAT
        text:=RND(0,13)
        CASE text OF
        WHEN 0,1,7,13
          combo:=TRUE
        OTHERWISE
          combo:=FALSE
        ENDCASE
      UNTIL combo=TRUE
    WHEN 9
      REPEAT
        text:=RND(1,15)
        CASE text OF
        WHEN 1,13,15
          combo:=TRUE
        OTHERWISE
          combo:=FALSE
        ENDCASE
      UNTIL combo=TRUE
    WHEN 10
      REPEAT
        text:=RND(0,11)
        CASE text OF
        WHEN 0,1,2,6,9,11
          combo:=TRUE
        OTHERWISE
          combo:=FALSE
        ENDCASE
      UNTIL combo=TRUE
    WHEN 13
      REPEAT
        text:=RND(0,11)
        CASE text OF
        WHEN 0,2,4,6,9,11
          combo:=TRUE
        OTHERWISE
          combo:=FALSE
        ENDCASE
      UNTIL combo=TRUE
    WHEN 14
      REPEAT
        text:=RND(0,9)
        CASE text OF
        WHEN 0,1,7,9
          combo:=TRUE
        OTHERWISE
          combo:=FALSE
        ENDCASE
      UNTIL combo=TRUE
    OTHERWISE
      backg:=1
      text:=6
    ENDCASE
  ENDCASE
  textbackground(backg)//background backg
  textborder(backg)//border backg
  textcolor(text)//pencolor text
ENDPROC change'colors
//
PROC c
  PRINT backg;text
ENDPROC c
//
PROC change'backg
  position:+1
  character$:=screen'line$(line)(position)
  CASE character$ OF
  WHEN "0","1","2","3","4","5","6","7","8","9"
    backg:=VAL(character$)
  WHEN "a","b","c","d","e","f"
    backg:=ORD(character$)-55
  OTHERWISE
    NULL
  ENDCASE
ENDPROC change'backg
//
PROC change'text
  position:+1
  character$:=screen'line$(line)(position)
  CASE character$ OF
  WHEN "0","1","2","3","4","5","6","7","8","9"
    text:=VAL(character$)
  WHEN "a","b","c","d","e","f"
    text:=ORD(character$)-55
  OTHERWISE
    NULL
  ENDCASE
ENDPROC change'text
```

more»

Message Board - continued

Example of text used by Southwest Missouri State University. This text is in a text file on *Today Disk 23*. The preceding article explains the use of @ # % + and =.

```
Updated 05-05-88
@#When will I be updated again?=
@+When I have more news!===
@#Welcome to the Industrial Technology
@+Electronic Bulletin Board.=
@++We're glad you're here!===
@#DON'T FORGET!=
@+Friday% May 6% is study day!=
@+Classes will not meet that day!==
@#FINALS SCHEDULE@=+
@+MONDAY@ all 11@00% all 12@30% and all
@12@00 except 12@00 Friday only classes=
@+TUESDAY@ all 10@00 and all 1@00.=
@+WEDNESDAY@ all 9@00% all 9:30%
@2@00 MWF% 2@00 MTWThF% and 4@00 TTh.=
@+THURSDAY@ all 8@00% 2@00 TTh and
@2@00 TThS% 4@00 MWF% and 4@00 MTWThF=
@+FRIDAY@ all 3@00% all 3:30%, and
@12@00 Friday only.============
@#NAIT NEWS!=
@+Congratulations to the new NAIT
@Officers@=
@+President---------Gerard Gregg=
@+Vice-President----Tom Reece=
@+Secty-Treasurer---James Meinsen=
@++We'll be depending on you!=======
@#IMPROVE YOUR EFFECTIVENESS=
@+Think about your activities for the
@day as you walk from place to place.=
@+Carry a pencil and note cards to
@capture ideas before they can get away.
@======
@#AIDD NEWS!=
@+Congratulations to the new AIDD
@Officers@=
@+President---------Fred Toomey=
@+Vice-President----Randy Green=
@+Secretary---------Armenta Richmond=
@+Treasurer---------Dan King
@++We'll be depending on you!=======
@#IMPROVE YOUR EFFECTIVENESS!
@+Plan personal time into your day.=
@Every day should include at least
@15 minutes doing something for
@yourself.=
@+Plan times for breaks as you work.=
@Your work quality and efficiency
@improve when you are refreshed.
@========
@#INTER-CLUB NEWS!=
@+Congratulations to the new Inter-Club
@Officers@=
@+Chairman---------Fred Toomey=
@+Secretary--------Gerard Gregg=
@++We'll be depending on you!=======
@#+Have a dull job to do?  =Clean your
@desk?  =Sort some papers?=
@++Work standing up. = Research shows
@you'll finish the job sooner.====
@$$end ■
```

Graph Sideways

by Robert E. Webb

This program was written for the C64 COMAL 2.0 and the MPS 801 Printer.

Graphs and Bar Charts are tools used to better understand how one quantity varies as a function of another. Large amounts of data can be presented as a simple two dimensional easily understood drawing. Sometimes it is advantageous to superimpose one or more graphs over another to see the relationships between two supposedly independent quantities. This program allows the easy formatting of a graph/bar chart, input of data, display of up to four graphs superimposed on each other, and screen dumps to a line printer in normal or sideways formats.

COMMAND SUMMARY

INPUT/EDIT DATA

- **Read disk file**
  Reads a previously stored data and format file from the disk for editing.
- **Edit scale format**
  Allows scale format information to be input or changed.
- **Edit data**
  Allows sample data to be input or changed.
- **Save to disk file**
  Saves the just edited scale format and data to the system disk, and returns control to the main menu.

PLOT GRAPH

Displays the disk directory.

- **File name 1**
  Inputs the name of the file from which the title, left scale, bottom scale, and data for the first graph will be taken.
- **Graph line or bar**
  Sets up for a line or bar graph.
- **Trace id character**
  If line graph is chosen, inputs the character which will be printed at each of the data points on the graph. Bars do not have an id character.
- **More files**
  Inputs whether a second graph is to be superimposed or not. If the input is 'yes', the above four steps will be repeated, up to three more times.
- **Right scale**
  Inputs whether there is to be a right scale or not. The right scale is used when any of the superimposed graphs has a different vertical unit and/or origin and full scale. The right scale format will be taken from the last file to be plotted.

PRINT GRAPH, NORMAL

Inputs the same information as PLOT GRAPH, but prints the graph in the normal text direction, using a 5 1/2 inch by 4 inch area on the paper.

PRINT GRAPH, SIDEWAYS

Inputs the same information as PLOT GRAPH but prints the graph at 90 degrees to the normal text mode using a 10 1/4 inch by 5 1/2 inch area on the paper. Seven example files are included, to demonstrate the use of the various options.

Graph demo 1.
Superimpose the four files "1985 kwh.grf", "1986 kwh.grf", "1987 kwh.grf", "1988 kwh.grf", as line graphs. Use id characters of 5, 6, 7, 8 respectively.

Graph demo 2.
Superimpose "nana dollars.grf" as a line graph over "nana sales.grf" as a bar graph.

Graph demo 3.
Superimpose "average kwh.grf" over "average temp.grf" as line graphs using K and T as identification characters.

more»

Graph Sideways - continued

```
dim'variables
REPEAT
  PAGE
  center("GRAPH SIDEWAYS",40)
  PRINT cr$,"0. Quit"
  PRINT "1. Input/Edit Data"
  PRINT "2. Plot Graph"
  PRINT "3. Print Graph, normal"
  PRINT "4. Print Graph, sideways"
  PRINT "5. Help"
  INPUT "Choice number? ": choice
  CASE choice OF
  WHEN 1
    edit'all
  WHEN 2
    p'flag:=0
    sideways:=0
    draw
  WHEN 3
    p'flag:=1
    sideways:=0
    draw
  WHEN 4
    p'flag:=1
    sideways:=1
    draw
  WHEN 5
    help
  OTHERWISE
    NULL
  ENDCASE
UNTIL choice=0
END "End of Program"
//
PROC edit'all
  REPEAT
    PAGE
    PRINT "1. Read disk file"
    PRINT "2. Edit scale format"
    PRINT "3. Edit data"
    PRINT "4. Save to disk file (exit to main
    menu)" //wrap line
    INPUT "Choice number? ": choice
    CASE choice OF
    WHEN 1
      DIR
      return'msg(0)
      edit'string("File name",data'file$(1))
      read'disk(data'file$(1))
    WHEN 2
      edit'scale
    WHEN 3
      edit'data
    OTHERWISE
      NULL
    ENDCASE
  UNTIL choice=4
  write'disk
ENDPROC edit'all
//
PROC center(string$,width)
  PRINT SPC$((width-LEN(string$))/2),string$
ENDPROC center
//
PROC dim'variables
  USE graphics
  textcolor(11)
  textbackground(12)
  textborder(12)
  sample'max:=200
  DIM top'line$ OF (80), smpl'id$ OF 7
  DIM next'line$ OF (40)
  DIM string$ OF 30, ch$ OF 1
  DIM data'file$(4) OF 20
  DIM strng$ OF 30, cr$ OF 1
  cr$:=CHR$(13)
  DIM sample$(sample'max) OF 5
  DIM title$ OF 30
  DIM h'unit$ OF 20
  DIM v'unit$ OF 20
  DIM trace'ch$(4) OF 1
  DIM bar$(4) OF 1
  DIM more'files$ OF 1, scale'right$ OF 1
  more'files$:="n"
  scale'right$:="n"
  FOR i:=1 TO 4 DO
    trace'ch$(i):="n"
    bar$(i):="l"
  ENDFOR i
  USE graphics
ENDPROC dim'variables
//
```

more»

```
PROC edit'scale
  PAGE
  return'msg(0)
  edit'string("Title",title$)
  edit'string("Horizontal unit",h'unit$)
  edit'string("Vertical unit",v'unit$)
  edit'number((h'unit$+" scale origin"),h'origin)
  edit'number((h'unit$+" full scale"),h'full)
  edit'number((h'unit$+" between ticks"),h'tics)
  edit'number(h'unit$+" between labels",h'label)
  edit'number((h'unit$+" sample interval"),h'inte
  rval) //wrap line
  edit'number((v'unit$+" scale origin "),v'origin)
  edit'number((v'unit$+" full scale"),v'full)
  edit'number((v'unit$+" between ticks"),v'tics)
  edit'number((v'unit$+" between labels"),v'label)
ENDPROC edit'scale
//
PROC edit'data
  PAGE
  return'msg(1)
  PRINT
  i:=0
  sample'id:=h'origin
  REPEAT
    i:=i+1
    smpl'id$:=STR$(sample'id)
    strng$:=("For "+h'unit$+" "+smpl'id
    $+"; "+v'unit$) //wrap line
    edit'string(strng$,sample$(i))
    sample'id:=sample'id+h'interval
  UNTIL sample$(i)="end"
ENDPROC edit'data
//
PROC edit'string(msg$,REF string$)
  PRINT msg$,"= ",string$,
  row:=(PEEK(214))+1
  col:=2+LEN(msg$)
  INPUT AT row,col: " ": new'string$
  string$:=new'string$
ENDPROC edit'string
//
PROC edit'number(msg$,REF nmbr)
  nmbstr$:=STR$(nmbr)
  edit'string(msg$,nmbstr$)
  nmbr:=VAL(nmbstr$)
ENDPROC edit'number
//
PROC write'disk
  return'msg(0)
  edit'string("File name",data'file$(1))
  OPEN FILE 2,"@0:"+data'file$(1),WRITE
  WRITE FILE 2: title$
  WRITE FILE 2: h'unit$
  WRITE FILE 2: v'unit$
  WRITE FILE 2: h'origin
  WRITE FILE 2: h'full
  WRITE FILE 2: h'tics
  WRITE FILE 2: h'label
  WRITE FILE 2: h'interval
  WRITE FILE 2: v'origin
  WRITE FILE 2: v'full
  WRITE FILE 2: v'tics
  WRITE FILE 2: v'label
  i:=1
  WHILE sample$(i)<>"end" DO
    WRITE FILE 2: sample$(i)
    i:=i+1
  ENDWHILE
  CLOSE FILE 2
ENDPROC write'disk
//
PROC read'disk(file'name$)
  FOR i:=1 TO sample'max DO
    sample$(i):="end"
  ENDFOR i
  OPEN FILE 2,file'name$,READ
  READ FILE 2: title$,h'unit$,v'unit$
  READ FILE 2: h'origin,h'full,h'tics,h'l
  abel,h'interval //wrap line
  READ FILE 2: v'origin,v'full,v'tics,v'label
  i:=0
  WHILE NOT EOF(2) DO
    i:=i+1
    READ FILE 2: sample$(i)
  ENDWHILE
  CLOSE FILE 2
ENDPROC read'disk
//
PROC left'scale
  IF sideways THEN
    textstyle(1,1,3,0)
```

more»

```
    id'line(3)
  ELSE
    textstyle(1,1,0,0)
    IF p'flag THEN
      id'line(1)
    ELSE
      id'line(2)
    ENDIF
  ENDIF
  y:=graph'v'pxls
  x:=(graph'h'pxls/2)-(LEN(title$)/2*8)
  splottext(x,y,title$)
  y:=-49
  x:=(graph'h'pxls/2)-(LEN(h'unit$)/2*8)
  splottext(x,y,h'unit$)
  //print vertical labels
  v'scale:=(graph'v'pxls)/(v'full-v'origin)
  i:=0
  labl:=0
  WHILE labl<(v'full-v'label) DO
    labl:=v'origin+(i*v'label)
    string$:=STR$(labl)
    IF LEN(string$)>4 THEN string$:=string$(1:4)
    x:=(-8)-((LEN(string$))*8)
    y:=(-4)+(i*v'label*v'scale)
    splottext(x,y,string$)
    i:=i+1
  ENDWHILE
  //Draw baselines
  smoveto(0,graph'v'pxls)
  sdrawto(0,0)
  sdrawto(graph'h'pxls,0)
  //Draw vertical tics
  i:=0
  x:=(-8)
  WHILE (i*v'tics)<(v'full-v'origin) DO
    y:=(i*v'tics*v'scale)
    smoveto(x,y)
    sdrawto((x+8),y)
    i:=i+1
  ENDWHILE
  x:=-48
  y:=((graph'v'pxls)/2)+(LEN(v'unit$)/2*8)-8
  plot'vert(v'unit$)
  //Plot horizontal labels
  h'scale:=(graph'h'pxls)/(h'full-h'origin+h'interval) //wrap line
  i:=0
  labl:=0
  WHILE labl<(h'full+h'interval-h'label) DO
    labl:=h'origin+(i*h'label)
    string$:=STR$(labl)
    IF LEN(string$)>4 THEN string$:=string$(1:4)
    y:=(-16)
    x:=(-4)+(i*h'label*h'scale)
    plot'vert(string$)
    i:=i+1
  ENDWHILE
  //Draw horizontal tics
  i:=0
  y:=-1
  WHILE (i*h'tics)<(h'full+h'interval-h'origin) DO
    x:=(i*h'tics*h'scale)
    smoveto(x,y)
    sdrawto(x,(y-6))
    i:=i+1
  ENDWHILE
ENDPROC left'scale
//
PROC plot'graph
  graphicscreen(0)
  window(scrn'lft,scrn'ryt,scrn'bot,scrn'top)
  clearscreen
  FOR fyle:=1 TO n'file DO
    read'disk(data'file$(fyle))
    h'range:=h'full-h'origin
    v'range:=v'full-v'origin
    h'scale:=(graph'h'pxls)/(h'range+h'interval)
    v'scale:=(graph'v'pxls)/(v'range)
    IF fyle=1 THEN
      left'scale
    ENDIF
    i:=1
    IF bar$(fyle)="b" THEN
      x:=0
      y:=0
      smoveto(x,y)
      i:=1
      WHILE sample$(i)<>"end" DO
        y:=((VAL(sample$(i))-v'origin)*v'scale)
        sdrawto(x,y)
        x:=x+(h'interval*h'scale)
```

more»

```
        sdrawto(x,y)
        y:=0
        sdrawto(x,y)
        i:=i+1
      ENDWHILE
    ELSE
      ch$:=trace'ch$(fyle)
      x:=0
      y:=((VAL(sample$(i))-v'origin)*v'scale)
      smoveto(x,y)
      i:=2
      WHILE sample$(i)<>"end" DO
        x:=x+(h'interval*h'scale)
        y:=((VAL(sample$(i))-v'origin)*v'scale)
        sdrawto(x,y)
        IF sideways THEN
          plottext(y-4,-x+4,ch$)
        ELSE
          textstyle(1,1,0,0)
          plottext((x-4),(y-4),ch$)
        ENDIF
        i:=i+1
      ENDWHILE
    ENDIF
  ENDFOR fyle
ENDPROC plot'graph
//
PROC plot'vert(string$)
  FOR i:=1 TO LEN(string$) DO
    splottext(x,y,(string$(i)))
    y:=y-8
  ENDFOR i
ENDPROC plot'vert
//
PROC draw
  clearscreen
  pencolor(11)
  background(12)
  border(12)
  n'file:=0
  PAGE
  DIR
  return'msg(0)
  REPEAT
    PRINT
    n'file:=n'file+1
    edit'string("File name "+STR$(n'file)
    +" ",data'file$(n'file)) //wrap line
    edit'string("Graph (l)ine or
    (b)ar",bar$(n'file)) //wrap line
    IF bar$(n'file)="b" THEN
      trace'ch$(n'file):="^"
    ELSE
      edit'string("Trace ID character,(n=
      none)",trace'ch$(n'file)) //wrap line
      IF trace'ch$(n'file)="n" THEN
        trace'ch$(n'file):=""
      ENDIF
    ENDIF
    edit'string("More files,(y)es/(n)o?",mor
    e'files$) //wrap line
  UNTIL (more'files$="n") OR (n'file=4)
  edit'string("Right scale,(y)es/(n)o?",scal
  e'right$) //wrap line
  IF sideways=0 THEN
    viewport(0,319,0,199)
    scrn'lft:=-49
    scrn'ryt:=270
    IF p'flag THEN
      scrn'top:=158
      scrn'bot:=-41
      graph'v'pxls:=159
    ELSE
      scrn'top:=149
      scrn'bot:=-50
      graph'v'pxls:=124
    ENDIF
    IF scale'right$="y" THEN
      graph'h'pxls:=220
    ELSE
      graph'h'pxls:=271
    ENDIF
    plot'graph
    IF scale'right$="y" THEN right'scale
    IF p'flag THEN
      printscreen("lp:",60)
      SELECT OUTPUT "lp:"
      center(h'unit$,80)
      SELECT OUTPUT "ds:"
    ELSE
      REPEAT
      UNTIL KEY$<>""
```

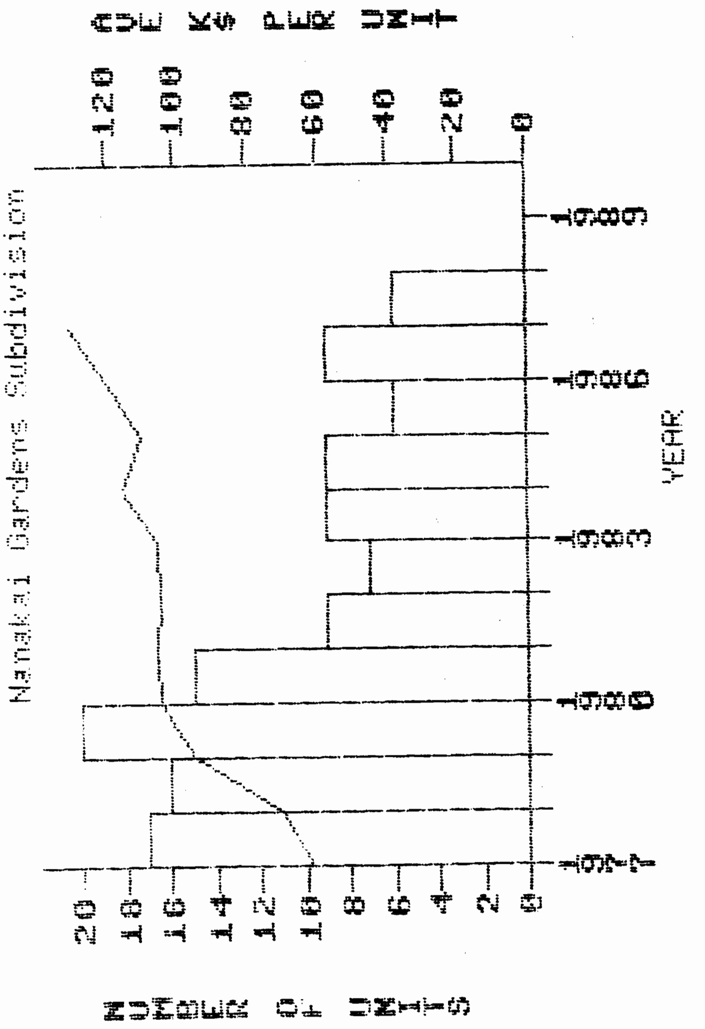
more»

Graph Sideways - continued

```
    ENDIF
  ELSE
    n'frames:=3
    IF scale'right$="y" THEN n'frames:=4
    viewport(0,319,5,199)
    graph'h'pxls:=536
    graph'v'pxls:=245
    FOR frame:=1 TO n'frames DO
      IF frame=1 THEN
        scrn'lft:=-50
        scrn'ryt:=269
        scrn'top:=49
        scrn'bot:=-145
      ELIF frame=2 THEN
        scrn'top:=-146
        scrn'bot:=-340
      ELIF frame=3 THEN
        scrn'top:=-341
        scrn'bot:=-535
      ELIF frame=4 THEN
        window(scrn'lft,scrn'ryt,-730,-536)
        right'scale
      ENDIF
      IF frame<>4 THEN plot'graph
      printscreen("lp:",60)
    ENDFOR frame
  ENDIF
  textscreen
ENDPROC draw
//
PROC splottext(x,y,text$)
  IF sideways THEN rotate(x,y)
  plottext(x,y,text$)
ENDPROC splottext
//
PROC smoveto(x,y)
  IF sideways THEN rotate(x,y)
  moveto(x,y)
ENDPROC smoveto
//
PROC sdrawto(x,y)
  IF sideways THEN rotate(x,y)
  drawto(x,y)
ENDPROC sdrawto
//
PROC rotate(REF x,REF y)
  temp:=y
  y:=-x
  x:=temp
ENDPROC rotate
//
PROC right'scale
  IF sideways THEN clearscreen
  smoveto(graph'h'pxls,0)
  sdrawto(graph'h'pxls,graph'v'pxls)
  //print vertical labels
  v'scale:=(graph'v'pxls)/(v'full-v'origin)
  i:=0
  labl:=0
  WHILE labl<(v'full-v'label) DO
    labl:=v'origin+(i*v'label)
    string$:=STR$(labl)
    IF LEN(string$)>4 THEN string$:=string$(1:4)
    x:=graph'h'pxls+9
    y:=(-4)+(i*v'label*v'scale)
    splottext(x,y,string$)
    i:=i+1
  ENDWHILE
  //Draw vertical tics
  i:=0
  x:=graph'h'pxls
  WHILE (i*v'tics)<(v'full-v'origin) DO
    y:=(i*v'tics*v'scale)
    smoveto(x,y)
    sdrawto((x+8),y)
    i:=i+1
  ENDWHILE
  x:=graph'h'pxls+42
  y:=((graph'v'pxls)/2)+(LEN(v'unit$)/2*8)-8
  plot'vert(v'unit$)
ENDPROC right'scale
//
PROC id'line(format)
  top'line$:=""
  next'line$:=""
  FOR i:=1 TO n'file DO
    ch$:=trace'ch$(i)
    IF ch$="" THEN ch$:="-"
    CASE i OF
    WHEN 1,2
      top'line$:=top'line$+SPC$(2)+ch$+"="+
      data'file$(i) //wrap line
```

more»

```
    WHEN 3,4
      next'line$:=next'line$+SPC$(2)+ch$+"="+
        data'file$(i) //wrap line
    ENDCASE
  ENDFOR i
  CASE format OF
  WHEN 1
    SELECT OUTPUT "lp:"
    center(top'line$,80)
    center(next'line$,80)
    center(title$,80)
    SELECT OUTPUT "ds:"
  WHEN 2
    x:=-49
    y:=graph'v'pxls+16
    top'line$:=SPC$((40-LEN(top'line$))/2)
    +top'line$ //wrap line
    splottext(x,y,top'line$)
    next'line$:=SPC$((40-LEN(next'line$))/2)
    +next'line$ //wrap line
    y:=graph'v'pxls+8
    splottext(x,y,next'line$)
  WHEN 3
    top'line$:=top'line$+next'line$
    top'line$:=SPC$((73-LEN(top'line$))/2)
    +top'line$ //wrap line
    x:=-39
    y:=graph'v'pxls+9
    splottext(x,y,top'line$)
  ENDCASE
ENDPROC id'line
//
PROC return'msg(flag)
  PRINT cr$,"* 'return'  accepts"
  PRINT "* New value, 'return' changes      "
  IF flag THEN PRINT "* Type 'end' to end
  data" //wrap line
  PRINT
ENDPROC return'msg
//
PROC help
  PAGE
  PRINT "Data can be plotted as lines or
  bars.",cr$ //wrap line
  PRINT "Four graphs can be superimposed; all"
  PRINT "must have same horizontal min/max
  units.",cr$ //wrap line
  PRINT "Data samples must be taken at"
  PRINT "regular intervals.",cr$
  PRINT "Tick labels are 4 chars maximum.",cr$
  PRINT "Title, left scale, bottom scale are"
  PRINT "plotted from the first file format.",cr$
  PRINT "The right scale, if used, is plotted"
  PRINT "from the last file format.",cr$
  INPUT "Through looking?": string$
ENDPROC help
```



Sets With String Elements

by Bill Inhelder

In *COMAL Today #13*, Joe Visser and Dick Klingens wrote an article and program dealing with set operations in COMAL 2.0. By treating the set operations as functions they are able to achieve both power and elegance in creating complex sequences of set operations.

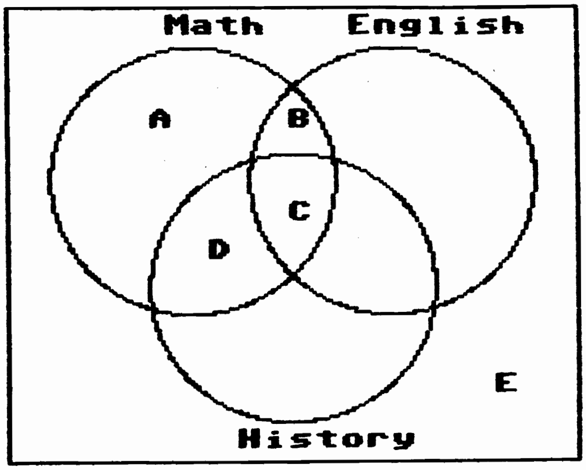
Unfortunately the elements of the sets are limited to the counting numbers. Since most set applications involve sets whose elements are string constants, I felt that it might be useful to modify and extend their program to involve sets with string elements.

By assigning a unique counting number to each string element, I was able to retain the structure and power of the original program. The entry of set elements and the construction of the sets was simplified by adding an input procedure. In addition the user may elect the demo mode which is useful in learning how to use the set operations and in determining their effect upon the sets. Alternatively the user may enter string elements for three sets together with an appropriate set of operations upon those sets.

The example given in the demo portion of the program involves an 11th grade physical education class of 30 students, 19 of whom take mathematics, 17 take English and 11 take history. The elements of the 3 sets are the first names of the students. Some of the students take two of the three subjects, some take all three and still others take only one. The situation is best illustrated by the Venn diagram shown in the next column.

Set1, representing the set of students taking mathematics, can be printed out using the command:

print elements(set1)



The number of elements is also printed out at the end of the set.

Those students taking both math and English can be determined with the command:

print elements(section(set1,set2))

The command to determine those students taking all three is:

print elements(section(section(set1,set2),set3))

To find those students taking math only (region A) is more complex:

print elements(minus(set1,union(section(
set1,set2),section(set1,set3)))) //wrap line

Verbally this is equivalent to those elements in **set1** but not in the union of the intersection of **set1 & 2** with the intersection of **set1 & 3**.

To determine the number of students who take none of the three subjects we subtract the number of students in the union of all three sets from 30. Thus:

print elements(union(union(set1,set2),set3))

more»

]

identifies 25 students, therefore region E contains 5 students.

The students in region B would be determined by those in the intersection of sets 1 and 2 but not in the intersection of all three.

New sets can be created by assigning the results of a set operation or operations to another set. Thus:

setx=section(set1,set2)

establishes a set containing the names of students in regions B and C. If:

sety:section(section(set1,set2),set3)

then the students in region B can be found by:

print elements(minus(setx,sety))

The original sets can be modified by adding or removing elements. Thus:

set1=addto(set1,"Jill")
or
set1:=remove(set1,"Jane")

The function element(set#) lists the indicated set to the printer. If a screen listing is desired delete line 3580.

A complete list of set operations and instructions for entry of string elements is given in the program string'set'calc.

Other set applications which the reader might try include:

1. lists of colors compatible with various screen background colors
2. word lists from a thesaurus for words of related meanings (eg. reticent, introverted and shy)
3. ingredients from three recipes.

In the 1st application the sets might be:

yellow background: black, red, brown, gray1, gray2
brown background: white, yellow, orange, lt.red, gray3
gray1 background: black, red, yellow, gray2, lt.green

Of course white and black backgrounds have considerably more compatible colors.

The following steps should be followed when entering your own sets:

1. Copy the list of elements to a sheet of paper. The first list will be numbered consecutively. In subsequent lists any element identical to an element in a previous list must bear the same number. Other elements may have any unique number up to 30.

2. Load string'set'calc. Determine what set of operations you wish to use with your sets. Enter them starting with line 3590. Remove remaining lines down to 3860 or enter the STOP and SELECT "ds:" commands on the next lines.

3. For screen display, remove line 3580.

4. Run the program. Follow the instructions for entry of set elements.

```
DIM binar$ OF 30, set1elmt$(30) OF 16
DIM set2elmt$(30) OF 16, jcount(30)
DIM masterset$(30) OF 16, set3elmt$(30) OF 16
//
FUNC bstr$(number) CLOSED
  DIM binar$ OF 30
  binar$:=bin2$(number)
  WHILE LEN(binar$)<30 DO binar$:="0"+binar$
  RETURN binar$
```

more»

```
  //
  FUNC bin2$(number)
    IF number=0 THEN
      RETURN ""
    ELSE
      RETURN bin2$(number DIV 2)+STR$(
      number MOD 2) //wrap line
    ENDIF
  ENDFUNC bin2$
  //
ENDFUNC bstr$
//
FUNC empty CLOSED
  IMPORT bstr$,bval //or a
  DIM binar$ OF 30 //simple
  binar$:=bstr$(0) //definition:
  RETURN bval(binar$) //RETURN 0
ENDFUNC empty
//
FUNC addto(set,elment$)
  found:=FALSE
  FOR j:=1 TO 30 DO
    IF masterset$(j)=elment$ THEN
      binar$:=bstr$(set)
      binar$(j):="1"
      found:=TRUE
      RETURN bval(binar$)
    ENDIF
  ENDFOR j
  IF found=FALSE THEN
    element:=maxno+1
    maxno:=element
    masterset$(element):=elment$
    binar$:=bstr$(set)
    binar$(element):="1"
    RETURN bval(binar$)
  ENDIF
ENDFUNC addto
//
FUNC bval(binar$) CLOSED
  IF binar$="" THEN
    RETURN 0
  ELSE
    l:=LEN(binar$)
    RETURN bval(binar$(1:l-1))*2+VAL(binar$(l))
  ENDIF
```

```
ENDFUNC bval
//
FUNC union(set1,set2) CLOSED
  IMPORT bstr$,bval
  DIM binar1$ OF 30, binar2$ OF 30
  binar1$:=bstr$(set1)
  binar2$:=bstr$(set2)
  FOR t:=1 TO 30 DO
    IF binar2$(t)="1" THEN binar1$(t):="1"
  ENDFOR t
  RETURN bval(binar1$)
ENDFUNC union
//
FUNC section(set1,set2) CLOSED
  IMPORT bstr$,bval
  DIM binar1$ OF 30, binar2$ OF 30
  DIM sect$ OF 30
  sect$:=bstr$(0)
  binar1$:=bstr$(set1)
  binar2$:=bstr$(set2)
  FOR t:=1 TO 30 DO
    IF binar1$(t)="1" AND binar2$(t)="1" THEN
      sect$(t):="1"
    ENDIF
  ENDFOR t
  RETURN bval(sect$)
ENDFUNC section
//
FUNC inset(set,elment$)
  FOR j:=1 TO 30 DO
    IF masterset$(j)=elment$ THEN element:=j
  ENDFOR j
  IF bstr$(set)(element:element)="1" THEN
    RETURN TRUE
  ELSE
    RETURN FALSE
  ENDIF
ENDFUNC inset
//
FUNC include(set,element) CLOSED
  IMPORT bstr$,bval
  DIM binar$ OF 30
  binar$:=bstr$(set)
  binar$(element):="1"
  RETURN bval(binar$)
ENDFUNC include
```

more»

```
//
FUNC remove(set,elment$)
  FOR j:=1 TO 30 DO
    IF masterset$(j)=elment$ THEN
      element:=j
    ENDIF
  ENDFOR j
  binar$:=bstr$(set)
  binar$(element):="0"
  RETURN bval(binar$)
ENDFUNC remove
//
FUNC elements(set)
  binar$:=bstr$(set)
  num:=0
  FOR t:=1 TO 30 DO
    IF binar$(t)="1" THEN
      IF masterset$(t)="" THEN
        NULL
      ELSE
        PRINT masterset$(t);
        num:+1
      ENDIF
    ENDIF
  ENDFOR t
  PRINT "#",
  RETURN num
ENDFUNC elements
//
FUNC minus(set1,set2) CLOSED
  IMPORT bstr$,bval
  DIM binar1$ OF 30, binar2$ OF 30
  binar1$:=bstr$(set1)
  binar2$:=bstr$(set2)
  FOR t:=1 TO 30 DO
    IF binar2$(t)="1" THEN binar1$(t):="0"
  ENDFOR t
  RETURN bval(binar1$)
ENDFUNC minus
//
FUNC symminus(set1,set2) CLOSED
  IMPORT bstr$,bval
  DIM binar1$ OF 30, binar2$ OF 30
  DIM min$ OF 30
  min$:=bstr$(0)
  binar1$:=bstr$(set1)
  binar2$:=bstr$(set2)
  FOR t:=1 TO 30 DO
    IF binar1$(t)="1" AND binar2$(t)="0" THEN
      min$(t):="1"
    ELIF binar1$(t)="0" AND binar2$(t)="1"
      min$(t):="1"
    ENDIF
  ENDFOR t
  RETURN bval(min$)
ENDFUNC symminus
//
FUNC i(set,e) CLOSED
  //          easy use of include
  IMPORT include
  RETURN include(set,e)
ENDFUNC i
//
PROC heading
  PAGE
  PRINT AT 11,16: "SET'CALC"
  PRINT AT 13,3: "Operations Upon String
  Element Sets" //wrap line
  PRINT AT 18,16: "Original Program By"
  PRINT AT 19,16: "J. Visser & D. Klingens"
  PRINT AT 21,16: "Modified Program Using"
  PRINT AT 22,16: "String Element Sets By"
  PRINT AT 23,16: "Bill Inhelder"
  FOR i:=1 TO 2500 DO NULL
  PAGE
ENDPROC heading
//
PROC instructions
  PAGE
  PRINT AT 7,1: "Starting with line 3590 you
  may begin" //wrap line
  PRINT "to write instructions to perform set"
  PRINT "operations on pairs of sets.The
  program" //wrap line
  PRINT "provides for a maximum of 3 sets for"
  PRINT "the user to enter. If the 2nd or 3rd "
  PRINT "set isn't needed, enter 0 for the"
  PRINT "mumber of elements. The total"
  PRINT "number of distinct elements in all 3"
  PRINT "sets must not exceed 30."
  PRINT
  PRINT "       Press any key to continue."
```

more»

```
  WHILE KEY$="" DO NULL
  PAGE
  PRINT "The following set operations are"
  PRINT "available to the user:"
  PRINT "1. elements(set#)-prints out the"
  PRINT "  elements in the numbered set"
  PRINT "2. remove(set#,string element)
   -removes" //wrap line
  PRINT "  the specified element from the set"
  PRINT "3. addto(set#,string element)-adds the"
  PRINT "  specified element to the set"
  PRINT "4. inset(set#,string element)-returns"
  PRINT "  true if in set; false if not in set"
  PRINT "5. union(setA,setB)-forms the union"
  PRINT "  of sets A and B"
  PRINT "6. section(setA,setB)-forms the inter-"
  PRINT "  section of A and B"
  PRINT "7. minus(setA,setB)-elements in A but"
  PRINT "  not in B, or vice versa (setB,
  setA)" //wrap line
  PRINT "8. symminus(setA,setB)-elements in A"
  PRINT "  but not in B or those in B but"
  PRINT "  not in A."
  PRINT
  PRINT "       Press any key to continue"
  WHILE KEY$="" DO NULL
  PAGE
  PRINT AT 6,1: "You will be given the option"
  PRINT "of running a demo or entering your"
  PRINT "own sets with appropriate"
  PRINT "instructions for operating on those"
  PRINT "sets. If you select the latter"
  PRINT "write the elements in each set on"
  PRINT "paper. Then enter the set operations"
  PRINT "starting with line 3590. Finally, run"
  PRINT "the program and input the elements"
  PRINT "of each set. The program will"
  PRINT "execute the group of instructions"
  PRINT "operating on your sets and output"
  PRINT "the results to the printer."
  PRINT
  PRINT "       Press any key to continue"
  WHILE KEY$="" DO NULL
  PAGE
  PRINT "Examples of set operations:"
  PRINT
  PRINT "print elements(section(set1,set2)) -"
  PRINT "outputs the set of elements common"
  PRINT "to both sets."
  PRINT
  PRINT "seta:=section(set1,set2) - assigns the"
  PRINT "set of elements common to both sets"
  PRINT "to seta."
  PRINT
  PRINT "To modify original sets:"
  PRINT "set1:=addto(set1,""element name"")"
  PRINT "set3:=remove(set3,""element name"")"
  PRINT
  PRINT "print elements(section(union(set1,set2)"
  PRINT ",set3)) - outputs the set of elements"
  PRINT "in the intersection of set3 with"
  PRINT "those in the union of sets1 & 2."
  PRINT
  PRINT "       Press any key to continue"
  WHILE KEY$="" DO NULL
  PAGE
ENDPROC instructions
//
PROC input'rtn
  USE system
  PAGE
  INPUT "Enter number of elements in set 1:":n
  FOR i:=1 TO n DO
    PRINT i,". ",
    INPUT set1elmt$(i)
    masterset$(i):=set1elmt$(i)
  ENDFOR i
  PRINT "             Constructing Set"
  set1:=empty
  FOR i:=1 TO n DO set1:=include(set1,i)
  pos:=currow-1
  PRINT AT pos,1: "For sets 2 and 3, any
  element identical" //wrap line
  PRINT "to one in set 1 or 2 MUST be"
  PRINT "assigned the same number. Other"
  PRINT "elements must have unique numbers"
  PRINT "different from the 1st or 2nd sets."
  INPUT "Enter number of elements in set 2:":m
  IF m<>0 THEN
    k:=1
    maxno:=0
    PRINT "number,string element:"
```

more»

```
    REPEAT
      INPUT j,set2elmt$(j)
      jcount(k):=j
      IF jcount(k)>maxno THEN maxno:=jc
      ount(k) //wrap line
      k:+1
    UNTIL k=m+1
    PRINT "            Constructing Set"
    FOR i:=1 TO m DO
      masterset$(jcount(i)):=set2elmt$(jcount(i))
    ENDFOR i
    set2:=empty
    FOR i:=1 TO m DO set2:=include(set2,j
    count(i)) //wrap line
  ENDIF
  k:=1
  pos:=currow-1
  INPUT AT pos,1: "Enter number of elements
  in set 3:": p //wrap line
  IF p<>0 THEN
    PRINT "number,string element:"
    REPEAT
      INPUT j,set3elmt$(j)
      jcount(k):=j
      IF jcount(k)>maxno THEN maxno:=jc
      ount(k) //wrap line
      k:+1
    UNTIL k=p+1
    FOR i:=1 TO p DO
      masterset$(jcount(i)):=set3elmt$(jcount(i))
    ENDFOR i
    set3:=empty
    FOR i:=1 TO p DO set3:=include(set3,jc
    ount(i)) //wrap line
  ENDIF
ENDPROC input'rtn
//
PROC read'rtn
  FOR i:=1 TO 19 DO
    READ set1elmt$(i)
    masterset$(i):=set1elmt$(i)
  ENDFOR i
  set1:=empty
  FOR i:=1 TO 19 DO set1:=include(set1,i)
  maxno:=0
  FOR i:=1 TO 17 DO
    READ j,set2elmt$(j)
    jcount(i):=j
    IF jcount(i)>maxno THEN maxno:=jcount(i)
  ENDFOR i
  FOR i:=1 TO 17 DO
    masterset$(jcount(i)):=set2elmt$(jcount(i))
  ENDFOR i
  set2:=empty
  FOR i:=1 TO 17 DO set2:=include(set2,jc
  ount(i)) //wrap line
  FOR i:=1 TO 11 DO
    READ j,set3elmt$(j)
    jcount(i):=j
    IF jcount(i)>maxno THEN maxno:=jcount(i)
  ENDFOR i
  FOR i:=1 TO 11 DO
    masterset$(jcount(i)):=set3elmt$(jcount(i))
  ENDFOR i
  set3:=empty
  FOR i:=1 TO 11 DO set3:=include(set3,jc
  ount(i)) //wrap line
  DATA "Samuel","Betty","Corine","Robert","Jack"
  DATA "Dorothy","Bill","Shirley","Paul"
  DATA "Heather","Jane","Lillian","Charles"
  DATA "David","Neville","Karen","Ruth"
  DATA "Thomas","Xavier"
  DATA 1,"Samuel",2,"Betty",3,"Corine"
  DATA 4,"Robert",5,"Jack",6,"Dorothy"
  DATA 7,"Bill",8,"Shirley",9,"Paul",10,"Heather"
  DATA 11,"Jane",12,"Lillian",20,"Peter"
  DATA 21,"Cynthia",22,"Lucille",23,"Richard"
  DATA 24,"Walter",13,"Charles",14,"David"
  DATA 15,"Neville",16,"Karen",17,"Ruth"
  DATA 6,"Dorothy",7,"Bill",20,"Peter"
  DATA 21,"Cynthia",22,"Lucille",25,"Frank"
ENDPROC read'rtn
//
PAGE
heading
INPUT "Do you wish instructions(y or n)?": q$
IF q$ IN "Yy" THEN instructions
PRINT "Want to enter your own sets(y or n)?"
INPUT q$
IF q$ IN "Yy" THEN
  input'rtn
ELSE
```

more»

Sets With String Elements - continued

```
  PRINT "          Constructing Sets"
  read'rtn
ENDIF
SELECT OUTPUT "lp:"
PRINT "Set of students taking math"
PRINT elements(set1)
PRINT "Set of students taking English"
PRINT elements(set2)
PRINT "Set of students taking history"
PRINT elements(set3)
PRINT "Set of students taking math & English"
PRINT elements(section(set1,set2))
PRINT "Set of students taking English &
 history" //wrap line
PRINT elements(section(set2,set3))
PRINT "Set of students taking math & history"
PRINT elements(section(set1,set3))
PRINT "Set of students taking all three"
PRINT elements(section(section(set1,set2),set3))
PRINT "Set of students taking only math"
PRINT elements(minus(set1,union(section(set1
,set2),section(set1,set3)))) //wrap line
PRINT "Set of all students-find the number"
PRINT "who are not taking any of the three."
PRINT elements(union(union(set1,set2),set3))
PRINT
set3:=addto(set3,"Jill")
set2:=remove(set2,"Lillian")
PRINT "Jill is added to set3"
PRINT elements(set3)
PRINT "Lillian is removed from set2"
PRINT elements(set2)
PRINT "Lillian is not removed from set1"
PRINT elements(set1)
SELECT OUTPUT "ds:"  ■
```

Program Size

Joel Rea provides a way to find a program size:

COMAL 0.14:

```
(peek(58)+256*peek(59))-peek(56)+256*(peek(57))
```

COMAL 2.0:

```
(peek($18)+$100*peek($19))-(peek(
$16)+$100*peek($17)) //wrap line  ■
```

Colorbook

by Dawn Hux

Artwork by Matthew Andrews, Kathi Dantley, Andre Dionne, Jeffrey Fortner, Michael Gibson, Andrew Holtom, Steven Kennedy, Steve McClay, Mark Method, Scott Mozingo, Leigh Shady, James Templeton, Paul Wallace

Last year I enjoyed teaching my first year COMAL students how to produce a computerized coloring book so I presented the program again to this year's beginning students. Each student designed his own title page, direction screen, picture and sprite. Many students produced excellent theme programs and I have included Steve McClay's program, "*hearts*", as an example. The other two coloring books are a composite of art work by students from two schools. To use these two programs load "*ccs color book*" or "*bbca color book*"

COMAL Today #19 carried an in depth discussion of the program so I will not repeat the details of the program here. It also explains how to design your own pictures and add them to the coloring book. Since we have used all of the available memory you will need to delete one or more of our pictures to make room for your art work. NOTE: You must load this program using the power driver version of COMAL which is on this disk. Older versions of COMAL do not have sufficient memory. ■

2.0 Function Key Tips

If you define F1, F3, or F5 remember that they can have two meanings (one set for turtle graphics)! If they don't work as expected, do a «*ctrl*»-U to switch meanings. This example with Superchip's files package lets you put the cursor at the start of a line displaying a text file you want to "see" (from a DIR). Press f1.

```
USE system
USE files
defkey(1,"type("13""11")"13"") ■
```

Expression Evaluator 0.14

by Lewis C. Brown

Ever since the META expression evaluator for COMAL 2.0 appeared in *COMAL Today #10*, I have been hoping to see a similar function or procedure for COMAL 0.14.

The Transactor (Vol. 8, No.6, pp 32-33, article by Paul Durrant) gives a machine language method for an expression evaluator in Basic. The method should work for COMAL 0.14 as well, but you would have to find all of the equivalent routines in the COMAL system.

In the meantime, here is a string expression evaluator for COMAL 0.14 that will work as long as the display screen is available. The method uses the screen in a way similar to that shown in the first COMAL 0.14 VAL function in *COMAL Today #1*, p.20, Jan-Feb, 1984.

The procedure loads the buffer with the required commands (using the keyboard buffer-fill procedure from *COMAL Today #6*, page 42), then executes a STOP within the procedure. The buffer is emptied and the buffer commands print the string on the screen with an assignment (y=), followed by a return, just as if you were typing the command in direct mode.

The numeric variable y now contains the value of the string. The buffer types out a CON command plus a return and the procedure finishes up.

If the string is written as a function of the variable x, then the string can be evaluated for any value assigned to x. A demo program on the disk illustrates this.

We can use this string expression evaluator while we are all waiting for a machine language programmer who knows the details of the COMAL 0.14 system well enough to provide us with a true META function for COMAL 0.14.

```
//Lewis C. Brown     Date: 050388
//Box 286, Rowayton, CT. 06853
//list "@0:expeval.v4.proc" //comal 0.14+
dim a$ of 40, b$ of 1, h$ of 1, c$ of 3, q(10)
print chr$(147)
//Demonstration-----
input "Enter function of x (such as 3*x+2):": a$
for x:=1 to 10 do
  expeval(a$)
  q(x):=y
endfor x
print chr$(147)
zone 3
for k:=1 to 10 do print "x=",k," ",c$,a$,"=",q(k)
print "Press a key...."
while key$=chr$(0) do null
print chr$(147)
input "Now enter a numeric expression: ": a$
expeval(a$)
print "That's all it does, folks!"
print "Changes a string expression into its
numerical equivalent!!!"//wrap line
print "If you want to look at the string,
enter""print a$"", then return"//wrap line
print "If you want to see the string value,
enter ""print y"", then return"//wrap line
end
//
proc expeval(a$)
  b$:=chr$(13); h$:=chr$(19); c$:="y= "
  print chr$(147) //clear screen
  print h$, //home
  print c$+a$,
  buffer(h$+b$+"con"+b$)
  stop
  print h$,
  print c$+a$+"= ";y
endproc expeval
//
proc buffer(string$) closed
  l:=len(string$) mod 11
  for x:=1 to l do
    poke 630+x,ord(string$(x))
  endfor x
  poke 198,l
endproc buffer ■
```

Pop Over Calculator

by David Warman

This program gives a demonstration of how the INTERRUPT command can be set to activate a calculator whenever he STOP key is pressed. This is an application of the popover system from COMAL Today #11. It can be useful in checkbook programs, budget programs, etc.

The STOP key is set up to cause an interrupt when pressed. After each line in the program is executed, an interrupt is checked for, and if one has occurred, control is transferred to the PROCedure named by the INTERRUPT command. When the PROCedure is finished, program flow returns to the line following where it left off. Since the STOP key is only checked AFTER a line is executed, some commands like INPUT, which hold program flow on a particular line until a certain condition is met, will prevent the interrupt from being handled immediately. For this reason, instead of INPUT, the PROCedure get'input is used to simulate an INPUT statement. A program that uses the calculator should also not have any one-line REPEATs, FORs, etc. that keep the program flow on one line when the user may be trying to access the calculator. The calculator PROCedure in this demo program has been shortened and will not work with the graphic screen. The full PROCedure proc.calculator is LISTed separately on the disk.

A couple of notes about the operation of the calculator: The DEL key is used like a clear-entry key on a calculator. Entering "4+3 DEL 4=" will result in 8, "5-2 DEL *3=" will display 15. Also, the square root function works on the result of the previous operation; "5+4 s" results in 3, not 7.

```
PROC calculator CLOSED //by David Warman
  // NOTE: The lines tagged with "//*" can be
  // deleted if the graphicscreen isn't used.
  // The following 3 lines should be early in
  // the program:
  //   first'call#:=TRUE
  //   calculator
  //   first'call#:=FALSE
  INTERRUPT
  USE graphics //*
  IMPORT first'call#
  textmode:=inq(13) //*
  graphmode:=inq(7) //*
  IF graphmode<2 THEN graphmode:=1-inq(14)//*
  TRAP ESC-
  IF first'call# THEN setup
  USE system
  DIM start'screen$ OF 1505
  DIM a$ OF 1, digits$ OF 11, number$ OF 15
  DIM operation$ OF 1, next'operation$ OF 1
  getscreen(start'screen$)
  IF NOT first'call# THEN
    IF NOT textmode THEN textscreen //*
    clear'keys
    popup
    setscreen(start'screen$)
    IF textmode THEN //*
      textscreen //*
    ELSE //*
      IF graphmode THEN //*
        fullscreen //*
      ELSE //*
        splitscreen //*
      ENDIF //*
    ENDIF //*
    clear'keys
  ENDIF
  INTERRUPT calculator
  //
  PROC popup
    col:=14; row:=2
    display'row:=row+1; display'col:=col+15
    CURSOR row,col
    PRINT AT 0,col: "-----------------"
    PRINT AT 0,col: "|               |"
    PRINT AT 0,col: "-----------------"
    PRINT AT 0,col: "|  7 8 9        |"
    PRINT AT 0,col: "|        /  CLR |"
    PRINT AT 0,col: "|  4 5 6 *  DEL |"
    PRINT AT 0,col: "|        -  (ce)|"
    PRINT AT 0,col: "|  1 2 3 +  "18"s"146"q
```

more»

```
  r  |" //wrap line
  PRINT AT 0,col: "|          =  ^    |"
  PRINT AT 0,col: "|  . 0        (pwr)|"
  PRINT AT 0,col: "-----------------"
  PRINT AT 0,col: "|   STOP to exit  |"
  PRINT AT 0,col: "-----------------"
  total:=0; a$:=""; digits$:=".0123456789"
  number$:=""; operation$:=""
  next'operation$:=""
  done:=FALSE
  REPEAT
    enter'number(number$,display'row,d
    isplay'col) //wrap line
    evaluate(number$)
  UNTIL done
ENDPROC popup
//
PROC clear'keys
  WHILE KEY$>"" DO NULL
  dummyesc:=ESC // clear stop key
ENDPROC clear'keys
//
PROC setup CLOSED
  TRAP ESC-
  FOR x#:=0 TO 12 DO
    READ byte#
    POKE $c86a+x#,byte#
  ENDFOR x#
  POKE $c7e2,$6a
  POKE $c7e3,$c8
  POKE $4d,PEEK($4d) BITOR $20
  DATA $a5,$4d,$29,$08,$f0,$06,$a9
  DATA $04,$05,$4d,$85,$4d,$60
ENDPROC setup
//
PROC enter'number(REF number$,r,c)
  a$:=""
  PRINT AT r,c-LEN(STR$(total))+1: total
  first'digit:=TRUE
  LOOP
    REPEAT
      a$:=KEY$
      IF ESC THEN
        done:=TRUE
        EXIT
      ENDIF
    UNTIL a$ IN digits$+"+-*/=s^"147"c"20""
    IF first'digit THEN PRINT AT r,c-14: S
    PC$(15) //wrap line
    first'digit:=FALSE
    IF a$ IN ""147"c" THEN
      PRINT AT r,c-14: SPC$(15)
      total:=0
      number$:=""
      operation$:="+"; next'operation$:=""
      PRINT AT r,c-LEN(STR$(total))+1: total
      PRINT AT r,c+2: " "
    ELIF a$=""20"" THEN
      PRINT AT r,c-14: SPC$(15)
      number$:=""
      PRINT AT r,c-LEN(STR$(total))+1: total
      first'digit:=TRUE
    ELIF a$ IN digits$ THEN
      IF LEN(number$)<12 THEN
        number$:+a$
        PRINT AT r,c-LEN(number$)+1: nu
        mber$ //wrap line
      ENDIF
    ELIF a$ IN "+-*/=s^" THEN
      next'operation$:=a$
      EXIT
    ENDIF
  ENDLOOP
  PRINT AT r,c-14: SPC$(15)
  PRINT AT r,c+2: next'operation$
ENDPROC enter'number
//
PROC evaluate(REF string$)
  IF number$="" THEN
    operation$:=next'operation$
  ELSE
    CASE operation$ OF
    WHEN "+"
      total:+VAL(number$)
    WHEN "-"
      total:-VAL(number$)
    WHEN "*"
      total:=total*VAL(number$)
    WHEN "/"
      total:=total/VAL(number$)
    WHEN "^"
      total:=total^VAL(number$)
```

more»

Pop Over Calculator - continued

```
      OTHERWISE
        total:=VAL(number$)
      ENDCASE
    ENDIF
    IF next'operation$="s" THEN
      total:=SQR(total)
      next'operation$:=""
    ENDIF
    operation$:=next'operation$
    number$:=""
  ENDPROC evaluate
  //
ENDPROC calculator
```

Use calls to the function get'input$ (listed below) to get input in programs that wish to have the pop over calculator:

```
FUNC get'input$(length) //by David Warman
  r:=currow; c:=curcol
  string$:=""
  LOOP
    REPEAT reply$:=inkey$ UNTIL reply$>""
    IF reply$=""13"" THEN
      EXIT
    ELIF reply$=""20"" THEN
      IF LEN(string$) THEN
        string$:=string$(1:LEN(string$)-1)
        PRINT AT r,c: string$+" "157"",
      ENDIF
    ELSE
      IF NOT reply$ IN unprintable$ AND LEN(
      string$)<length THEN //wrap line
        string$:+reply$
        PRINT AT r,c: string$,
        IF LEN(string$)=length THEN PRINT
        ""157"", //wrap line
      ENDIF
    ENDIF
  ENDLOOP
  RETURN string$
ENDFUNC get'input$ ■
```

Extra

by David Warman

In addition to the Pop Over Calculator program just presented, I've also written a PROCedure call "proc.prompt". It will display a non-destructive message anywhere on the screen, wait for the user to enter input or for a certain amount of time, then will erase the message and re-display the original text, similar to the way the COMAL error messages work. There is a slightly different version of prompt, for COMAL 2.0 and COMAL 0.14/Power Driver. Both are presented below:

COMAL 2.0 version

```
PROC prompt(r,c,message$) CLOSED
  USE system
  DIM text$ OF LEN(message$)
  rr:=currow; cc:=curcol
  CURSOR r,c
  OPEN FILE 3,"ds:"
  INPUT FILE 3: text$
  CLOSE FILE 3
  PRINT AT r,c: message$
  WHILE KEY$="" DO NULL //any kind of
  //input can be waited for in above line
  CURSOR r,c
  PRINT text$,
  CURSOR rr,cc
ENDPROC prompt
```

COMAL 0.14/Power Driver version

```
PROC prompt(r,c,message$) CLOSED
  DIM text$ OF LEN(message$)
  rr:=CURROW; cc:=CURCOL//curcol
  CURSOR r,c
  OPEN FILE 3,"",UNIT 3,READ
  INPUT FILE 3: text$
  CLOSE FILE 3
  PRINT AT r,c:message$
  WHILE KEY$=CHR$(0) DO NULL //any kind
  //of input can be waited for in above line
  CURSOR r,c
  PRINT text$,
  CURSOR rr,cc
ENDPROC prompt ■
```

Graphing Parametric Equations

by Bill Inhelder

In High School, parametric equations are first introduced in 2nd year algebra, introductory analysis and pre-calculus classes. The usual treatment involves converting equations from parametric to rectangular coordinate form, developing parametric equations and some graphing. Because of the complexity, graphing is limited to a few simple equations.

Param'graph is useful for both students and teachers. It graphs both parametric equations and the resultant graph on the same screen and in such a manner as to reinforce the concept of parametric equations and the elimination of the parameter. Three windows are used to achieve this objective. The first window, in the upper left hand portion of the screen, is used to graph the function **y=g(t)**, where **t** is the independent variable on the horizontal axis and **y** is the dependent variable on the vertical axis. The second window, located in the lower right hand portion of the screen, is used to graph **x=f(t)** (rotated 90 degrees), where **t**, the independent variable, is on the vertical axis and **x**, the dependent variable, is on the horizontal axis. The third window, located in the upper right hand portion of the screen, is used to graph the relation **(f(t),g(t))** with **f(t)** on the horizontal axis and **g(t)** on the vertical axis. Thus **g(t)** is carried across horizontally to become the **y** value of the point while **f(t)** is carried up vertically to become the **x** value of the point. For each value of **t** the points in the respective windows are plotted thus permitting the student to monitor the process. A delay loop might be added if the teacher wishes to demonstrate this process to the class.

Several other features of the program are significant. To expedite entry of the parametric equations, the non-rommed version of **pkg.meta** was linked to the program. The user is offered the options of automatic scaling or user-determined scaling.

With automatic scaling **(-3.14<=t<=3.14)** unity is preserved on all screen axes; that is, one unit on the horizontal axis is equal in length to one unit on the vertical axis. Thus shape is preserved so that a circle will appear on the screen as a circle and not as an ellipse. Unfortunately this form of automatic scaling is not appropriate for all parametric equations because the resultant graph and/or the graphs of the parametric equations may run beyond their windows. Hence the need for the user-defined scaling option.

In the user-defined scaling option, if the **max** and **min** values of **t** are such that the **ABS(min)=max** then unity will be preserved on the axes of the resultant graph only. Generally unity on the other two graphs will be different from one axis to the other and different from the axes of the resultant graph. If **ABS(min)<>max** then unity may be lost on all the axes. Thus the shape of the resultant graph may be distorted. Some parametric equations may demand such an unbalanced condition (see trajectory problem). The general rule in user-defined scaling is to enter a balanced condition for **t**. If the resultant graph runs beyond the window, increase **t** until it is within the window. Parametric equation graphs which run beyond their windows can be corrected by increasing the **y** values. Explicit instructions for the user-defined scaling are included in the program.

The following parametric equations may be of some interest. Automatic scaling is implied unless otherwise indicated. For user-defined scaling, the first two numbers represent the minimum and maximum values for **t** (the horizontal axis) and the last two numbers are the minimum and maximum values for the output of both **f(t)** and **g(t)**.

more»

1.	x=cos(t)
	y=sin(t)
		unit circle

2.	x=1-cos(t)
	y=1+sin(t)
		displaced circle

3.	x=sin(t)
	y=2*cos(t)
		vertical ellipse

4.	x=cos(t)
	y=sin(t)^2
		parabola, y>=0

5.	x=(1-cos(t))*cos(t)
	y=(1-cos(t))*sin(t)
		cardioid

6.	x=2*cos(t)^3
	y=2*sin(t)^3
		hypocycloid

7.	x=t-1.5*sin(t)
	y=1-1.5*cos(t)
		prolate cycloid
		(-2,8;-3,3)

8.	x=t-sin(t)
	y=1-cos(t)
		cycloid
		(-7,7;-5,5)

9.	x=t*t
	y=t*t*t
		semi-cubical parabola

10.	x=(1-2*cos(t))*cos(t)
	y=(1-2*cos(t))*sin(t)
		Limacon of Pascal

11.	x=2*sin(3*t)*cos(t)
	y=2*sin(3*t)*sin(t)
		3-leaved rose

12.	x=cos(2*t)
	y=cos(2*t)*tan(t)
		strophoid

13.	x=sin(2*t)
	y=cos(3*t)
		Lissajous figure

14.	x=sin(t)^2
	y=cos(t)^2
		x+y=1 where x & y >=0

15.	x=.5*tan(t)
	y=1.5*sin(3*t)
		
		-1.5,1.5;-5,5

For a practical application consider the parametric equations for a trajectory problem. First the equations where frictional force is neglected:

```
x=v*t*cos(A)
y=v*t*sin(A)-(g*t^2)/2
```

where v is the initial velocity, A is the angle of elevation in radians and g=32 ft/sec squared. Note that the independent variable (parameter) is t for time in seconds.

Specifically consider throwing a baseball at 120ft/sec at an angle of 40 degrees or .698 radians. The equations become:

```
x=120*t*cos(.689)
y=120*t*sin(.689)-16*t^2
```

scaling values are:

```
tmin=0, tmax=4.8
```

The graph shows a range of 441 feet and a height of 93 feet. Note that the graph is a parabola.

Next, for greater realism, frictional force is introduced which is proportional to the velocity. The equations become:

```
d=(1-exp(-k*t/m))
x=m/k*(v*cos(A))*d
y=m/k*((g*m/k+v*sin(A))*d-g*t)
```

where m is the mass (in slugs) of the projectile and k is the constant of proportionality. Again time t is the only independent variable.

Specifically consider throwing a baseball (1/3 lb or about 1/96 slugs) at 120 ft/sec at an angle of 40 degrees where k=.0021. Thus m/k is approximately equal to 5 and k/m is approximately equal to 0.2. The equations now simplify to:

```
x=5*(120*cos(.689)*(1-exp(-.2*t))
y=5*((160+120*sin(.689))*(1-exp(-.2*t))-32*t)
```

which further simplifies to

```
x=465.6*(1-exp(-.2*t))
y=1185.68*(1-exp(-.2*t))-160*t
```

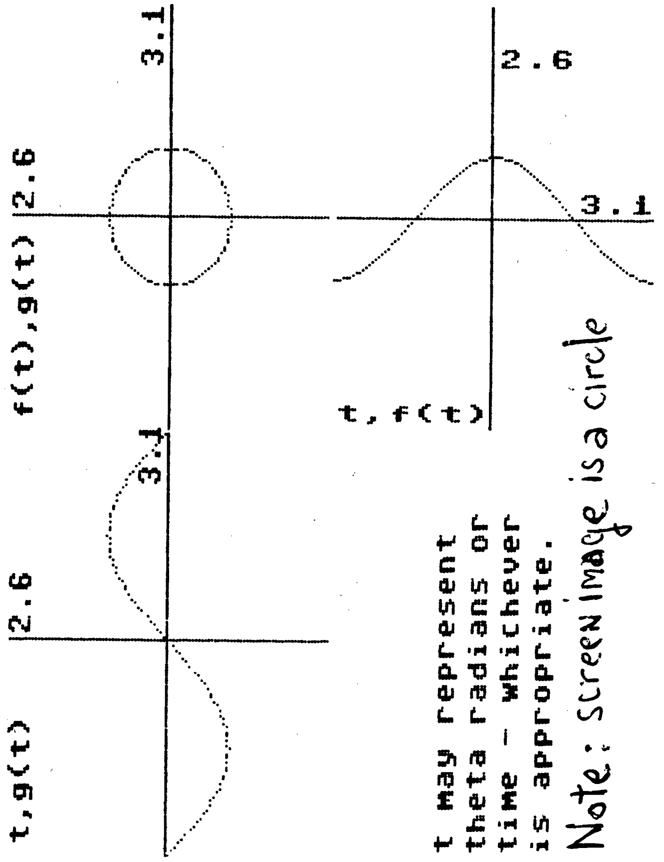
The scaling values are:

```
tmin=0, tmax=4.2
vertmin=-150, vertmax=150
```

more»

Graphing Parametric Equations - continued

The graph, which is no longer a parabola, illustrates how the range is shortened and the height is lowered due to friction. The new range is 264 feet and the new height 71 feet. <u>K</u> is probably too large for this small mass; however, the lack of symmetry of the graph shows up better with the <u>k</u> which was used.



t may represent theta radians or time - whichever is appropriate.

Note: Screen image is a circle

```
PAGE
DIM xt(101),yt(101),expr1$ OF 40,expr2$ OF 40
USE meta
USE graphics
heading
oncemore$:="n"
WHILE oncemore$="n" DO
  instructions
  oncemore$:="a"
  WHILE oncemore$="a" DO
    scaling
    calculations
    background(1)
    pencolor(0)
    window'setup
    graph'rtn
    viewport(0,319,0,199)
    window(0,319,0,199)
    clearscreen
    textscreen
    PAGE
  ENDWHILE
ENDWHILE
END
//
PROC heading
  PRINT AT 12,5: "Graphing Parametric
  Equations" //wrap line
  PRINT AT 15,20: "By Bill Inhelder"
  FOR i:=1 TO 1500 DO NULL
  PAGE
ENDPROC heading
//
PROC instructions
  PRINT AT 3,1: "Given a pair of parametric
  equations," //wrap line
  PRINT "x=f(t) and y=g(t), this program will"
  PRINT "graph each equation and the curve"
  PRINT "which results from the elimination of"
  PRINT "the parameter."
  PRINT
  PRINT "The option of automatic scaling is"
  PRINT "available (ie -3.14 to 3.14 for the"
  PRINT "horizontal axis and an appropriate "
  PRINT "vertical axis in order to preserve"
  PRINT "unity on both axes). Since this scale"
  PRINT "is not appropriate for all parametric"
  PRINT "equations, the user may select"
  PRINT "whatever is appropriate. However the"
  PRINT "scaling of the axes may no longer be"
  PRINT "equivalent."
  PRINT "Once graphing is complete, press 'a' "
```

more»

```
  PRINT "to run again with a different scaling"
  PRINT "or 'n' to run again with a different"
  PRINT "equation set or 'q' to quit."
  PRINT
  PRINT "         Press c to continue"
  REPEAT
    cc$:=KEY$
  UNTIL cc$ IN "cC"
  PAGE
  PRINT AT 5,1: "The program will temporarily"
  PRINT "halt to permit entry of parametric"
  PRINT "equations. When prompt appears, type"
  PRINT "equations to the right of the = sign"
  PRINT "using the parameter t (ie t-sin(t)), "
  PRINT "then press RETURN."
  PRINT
  PRINT
  PRINT
  INPUT "f(t)=": expr1$
  INPUT "g(t)=": expr2$
  PAGE
ENDPROC instructions
//
PROC scaling
  INPUT "Do you wish automatic scaling(y or
  n):": ans$ //wrap line
  IF ans$ IN "yY" THEN
    xmax:=3.14; xmin:=-3.14
    ymax:=.63*xmax; ymin:=-.63*xmax
  ELSE
    PRINT
    PRINT
    PRINT "Only the vertical axis will move"
    PRINT "left or right within the window."
    PRINT "Therefore the following input"
    PRINT "conditions apply:"
    PRINT " 1. min x value <= 0"
    PRINT " 2. max value > 0"
    PRINT " 3. min y value < 0 and max y
    value > 0" //wrap line
    PRINT " 4. abs(min y value)=max y value
    (ex." //wrap line
    PRINT "    abs(-4)=4)."
    PRINT
    INPUT "Minimum x value:": xmin
    INPUT "Maximum x value:": xmax
    REPEAT
      INPUT "Minimum y value:": ymin
      INPUT "Maximum y value:": ymax
    UNTIL ABS(ymin)=ymax
  ENDIF
  xmaxrandt:=INT(10*(10*xmax+.5)/10)/10
  ymaxrandt:=INT(10*(10*ymax*1.37+.5)/10)/10
  strmaxx$:=STR$(xmaxrandt)
  strmaxy$:=STR$(ymaxrandt)
  xrange:=xmax-xmin
ENDPROC scaling
//
FUNC f(t)
  eval("x="+expr1$)
  RETURN x
ENDFUNC f
//
FUNC g(t)
  eval("y="+expr2$)
  RETURN y
ENDFUNC g
//
PROC calculations
  PAGE
  t:=xmin
  PRINT AT 12,11: "CALCULATING VALUES"
  xtmax:=-1000
  xtmin:=1000
  ytmax:=-1000
  ytmin:=1000
  FOR i:=1 TO 101 DO
    xt(i):=.76*f(t)
    IF xt(i)>xtmax THEN xtmax:=xt(i)
    IF xt(i)<xtmin THEN xtmin:=xt(i)
    yt(i):=.76*g(t)
    IF yt(i)>ytmax THEN ytmax:=yt(i)
    IF yt(i)<ytmin THEN ytmin:=yt(i)
    t:=t+xrange/100
  ENDFOR i
ENDPROC calculations
//
PROC window'setup
  graphicscreen(0)
  viewport(0,159,100,199)
  moveto(0,150)
  drawto(157,150)
```

more»

```
  plottext(138,152,strmaxx$)
  m:=ABS(xmin)/xrange*160
  IF ABS(xmin)<>xmax THEN
    moveto(m,100)
    drawto(m,199)
    plottext(m+2,192,strmaxy$)
    plottext(2,103,"t,g(t)")
  ELSE
    moveto(80,100)
    drawto(80,199)
    plottext(82,192,strmaxy$)
    plottext(4,192,"t,g(t)")
  ENDIF
  viewport(160,319,0,100)
  m:=100-ABS(xmin)/xrange*100
  IF ABS(xmin)<>xmax THEN
    moveto(240,0)
    drawto(240,100)
    moveto(170,m)
    drawto(310,m)
    textstyle(1,1,3,0)
    plottext(310,m-2,strmaxy$)
    plottext(242,10,strmaxx$)
    plottext(168,92,"t,f(t)")
    textstyle(1,1,0,1)
  ELSE
    moveto(160,50)
    drawto(319,50)
    textstyle(1,1,3,0)
    plottext(242,23,strmaxx$)
    textstyle(1,1,0,1)
    moveto(240,0)
    drawto(240,98)
    textstyle(1,1,3,0)
    plottext(296,48,strmaxy$)
    plottext(162,98,"t,f(t)")
    textstyle(1,1,0,1)
  ENDIF
  viewport(160,319,101,199)
  IF ABS(xmin)<>xmax THEN
    xm:=1.316*xtmax
  ELSE
    xm:=xmax
  ENDIF
  IF ABS(xmin)<>xmax THEN
    xn:=1.316*xtmin
  ELSE
    xn:=xmin
  ENDIF
  xrange3:=xm-xn
  xmaxrandt:=INT(10*(10*xm+.5)/10)/10
  IF xtmax>4*xmax THEN
    ymaxrandt:=INT(10*(10*xtmax*99/159+.5
    )/10)/10 //wrap line
  ELIF ABS(xmin)<>xmax THEN
    ymaxrandt:=INT(10*(10*xmax*99/159+.5
    )/10)/10 //wrap line
  ELSE
    ymaxrandt:=INT(10*(10*xmax*.87+.5)/10)/10
  ENDIF
  strmaxx$:=STR$(xmaxrandt)
  strmaxy$:=STR$(ymaxrandt)
  moveto(160,150)
  drawto(319,150)
  plottext(296,152,strmaxx$)
  m:=ABS(xn)/xrange3*160+160
  IF ABS(xn)<>xm THEN
    moveto(m,100)
    drawto(m,199)
    plottext(m+2,192,strmaxy$)
    plottext(170,103,"f(t),g(t)")
  ELSE
    moveto(240,100)
    drawto(240,199)
    plottext(242,192,strmaxy$)
    plottext(162,192,"f(t),g(t)")
  ENDIF
  viewport(0,158,0,98)
  plottext(2,60,"t may represent")
  plottext(2,50,"theta radians or")
  plottext(2,40,"time - whichever")
  plottext(2,30,"is appropriate.")
ENDPROC window'setup
//
PROC graph'rtn
  t:=xmin
  FOR i:=1 TO 101 DO
    viewport(0,159,100,199)
    window(xmin,xmax,ymin,ymax)
    plot(t,yt(i))
    t:=t+xrange/100
    viewport(160,319,0,100)
```

more»

Graphing Parametric Equations - continued

```
    window(ymin,ymax,xmax,xmin)
    IF ABS(xmin)<>xmax THEN
      plot(.82*xt(i),t)
    ELSE
      plot(.76*xt(i),t)
    ENDIF
    viewport(160,319,101,199)
    IF xtmax>4*xmax THEN
      window(1.316*xtmin,1.316*xtmax,-.48
      *xtmax,.48*xtmax) //wrap line
    ELIF ABS(xmin)<>xmax THEN
      window(xn,xm,-xmax*99/159*.73,xmax
      *99/159*.73) //wrap line
    ELSE
      window(xn,xm,.62*xmin,.62*xmax)
    ENDIF
    plot(xt(i)*1.316,yt(i))
  ENDFOR i
  REPEAT
    oncemore$:=KEY$
  UNTIL oncemore$ IN "AaNnQq"
  PAGE
ENDPROC graph'rtn
//
PROC link'meta CLOSED
  USE system
  setpage($76)
  DIM task$ OF 25
  IF PEEK($8014)<>4 THEN
    PAGE
    task$:="LINK ""pkg.meta"""13""
    task$:+"RUN "11""13""
    FOR x#:=1 TO LEN(task$) DO
      POKE 49151+x#,ORD(task$(x#))
    ENDFOR x#
    POKE $c866,$00
    POKE $c867,$c0
    POKE $c865,LEN(task$)
    STOP
  ENDIF
ENDPROC link'meta ■
```

Catalog DB

by Paul Keck

Free'catalog'db is an organizer for addresses of places which offer free or almost free catalogs. It prints mailing labels on standard stock. Scrolling is done with the cursor keys, T for top, and B for bottom. A couple of notes-

1. The Dept. and Box categories automatically print a Dept. or Box in front of the string for this field. To override this, an * in the first column is used. Example: if '12-B' is the text, 'Dept. 12-B' would be printed; if '*12-B', then '12-B' would be printed.

2. I used some initials as shorthand in the comments field. T means send your title, C your company, and O your organization. Some places request that information, since they like to do business with people with expense accounts.

3. The 'print custom label' option will let you print five lines of text on a label. This would normally be stuff like 'I would like info on such-and-such.' To just ask for a free catalog, select 'print request'.

4. Marking entries allows you to go through and mark all the ones you want printed, and do them all at once. 'A' is the top one, 'B' is the middle one, and 'C' is the bottom one. ■

VALUE Function for 2.0

Here is a value function that returns 0 for non-numeric strings (like Power Driver):

```
FUNC value(a$) CLOSED
  TRAP
    RETURN VAL(a$)
  HANDLER
    RETURN 0
  ENDTRAP
ENDFUNC value ■
```

Cryptograms

by David Warman

A cyptogram is a puzzle consisting of words in which each letter has been replaced by another letter. For example, the phrase:

To be or not to be

might be encoded like this:

Bx qa xl cxb bx qa

All **t**'s are replaced by **b**'s, all **o**'s by **x**'s, etc. To solve the puzzle, you must figure out which letters have been substituted for which. These hints might help.

- **E** is the most frequently used letter in the English language
- Other common letters are **s**, **l**, **r**, **s**, **t**, and the other vowels
- Common word endings are **-ed** and **-ing**
- A single letter following an apostrophe is usually **s** or **t**
- **The** is a very common word

When the computer asks what letter to replace, type the letter you want to change, then the letter you want to change it to. To erase a guess, type SPACE as the letter to replace with.

When more than one person is playing, the same puzzle will be given to each player, and the solving time will be recorded for each player. Of course, while one person is playing, those who haven't played yet should leave the room. Solving times are accumulated from one puzzle to the next.

While playing, press **f1** to see each player's total time, or press **f8** to give up. Giving up adds 15 minutes to your time.

This Power Driver version of Cryptograms contains all of the features of the 2.0 version, but because of the reduced memory, fewer puzzles are permitted in memory at once. This program can read the longer data files used with version 2.0, but the file read will stop before memory is over-filled. If you later alter the puzzles and attempt to save the file back to the same disk, the original file will be replaced by the shorter one. The program will warn you before you do this. [*Note, if you compile the program, you will gain about 10K more free memory.*]

Here are the main menu options:

1. **Play Game:** Obviously, this starts the game. "Cryptograms" lets up to 6 players play against each other, taking turns solving the same puzzle and competing for the shortest solving time. Times are kept by the clock package (in version 2.0) which is LINKed to the program. A running time is displayed at the top of the screen while you are solving the cryptogram.

2. **Make puzzle:** This option allows you to type in your own cryptograms, which can be saved to disk. Immediately after the program starts, it asks for the name of a file from which to load the cryptograms. You can choose the default file, which contains several ready-to-play puzzles, or you can start a file of your own puzzles. Note: if you are not using the disk-loaded rabbit package, you can increase the "most'puzzles'allowed" variable in the "init" PROCedure to allow more puzzles in memory and in a single file.

3. **Edit/erase puzzles:** This option lets you correct any mistakes you may have made when first typing in a cryptogram. (It can also be used to cheat by previewing all of the puzzles in the current file.)

4. **Quit:** This saves any new or altered puzzles, then terminates the program. ■

ORDER FORM Subscriber#__________

Name:__________

Street:__________

City/St/Zip:__________

Visa/MC#:__________

Exp Date:______ Signature:__________

Sept'88-Prices subject to change without notice

TO ORDER:

- Pull out this 4 page section
- Fill in subscriber# / address (above)
- Check [x] each item you want to order
- Add up total for items (fill in below)
- Add shipping/handling (fill in below)
- Send check/money order for Grand Total
  -- OR -- Fill in charge info (above) ...
  we will calculate the total & shipping
- Mail to: COMAL Users Group, U.S.A, Ltd,
  5501 Groveland, Madison, WI 53716
  ... or call 608-222-4432

SUBSCRIPTIONS:

Expired subscribers must renew before they may order at subscriber prices (renewal starts with the issue where you left off). New subscribers can order at the same time as subscribing.

[ ] $4 per issue subscription/renewal
(Canada/APO add $1 per issue, 1st Class)
[ ] $9 per disk subscription/renewal

[ ] $4.95 each backissue; circle issues wanted:
1 2 3 4 5 6 7 8 9 10 11 12 13
14 15 16 17 18 19 20/21 22 23
[ ] $19.95 *COMAL Yesterday* (spriral bound #1-4)

NOTICE: Minimum order $10. Shipping is extra: $3 minimum shipping; $3 per book (small books less); Canada, APO & 1st Class add $1 more per book and newsletter issue. Newsletter is published as time permits; size and format varies. Cancelled subscriptions receive a partial credit; no money back. All orders must be prepaid in US Dollars; Canada and USA only. Prices shown are subscriber prices and reflect a $2 discount. Allow 2 weeks for checks to clear. $15 charge for checks not honored by the bank. Wisconsin residents add 5% sales tax.

ENTER TOTALS HERE:

Item Total ($10 minimum) $__________
Ship Total ($3 minimum): $__________
Grand Total enclosed: $__________

SYSTEMS:

[ ] **C64 Power Box**db
Includes 3 utility disks, a toolbox of about 250 procedures and functions and a free compiler. $29.95 + $4 ship
[ ] **Starter Kit Option**: 12 issues of COMAL Today, 56 page index, 2 books and 5 disks! Over 1,000 pages! $29.95 + $4 ship
[ ] **C64 Keyboard Overlay Option**: excellent condensed command reference. $2.50
[ ] **Option** ... $7.95 Doc Box

[ ] **C64 COMAL 2.0 Cart Complete***
Tutorials, references, packages, ... with all 5 options below: $179.95+$7 ship
[ ] **C64 COMAL 2.0 Cartridge***
64K cartridge with empty socket for up to 32K EPROM. Plain, no documentation $99.95
[ ] **Deluxe Option**db: three books: *2.0 Keywords*, *Cart Graphics & Sound*, *Common COMAL Reference*; 4 cart demo disks. $29.95+$4 ship
[ ] **Packages Option**db: three books: *COMAL 2.0 Packages*, *Packages Library 1* (17 packages), *Packages Library 2* (24 packages); Superchip on Disk (9 packages) $36.95 + $4 ship
[ ] **Applications/Tutorial Option**db: three books: *COMAL Collage*, *3 Programs In Detail*, *Graph Paper*. $36.95 + $4 ship
[ ] **Option** ... $7.95 Doc Box

[ ] **Super Chip**: plug in chip for C64 cart; adds about 100 commands to the cartridge. $24.95

[ ] **CP/M COMAL 2.10**
Full COMAL system disk plus the DEMO disk, packed in a Doc Box with manual. Works in C128 CP/M mode. $49.95 + $4 ship
[ ] **Compiler Option**: RUNTIME system ... $9.95
[ ] **C128 Graphics Option**: Package disk $9.95
[ ] **CP/M Package Guide Option**: Reference on how to write packages $14.95 + $2 ship
[ ] **Option**: Common COMAL Reference $16.95

[ ] **UniComal IBM PC COMAL 2.1***
Full fast system, with extensive reference & tutorial packed in a Doc Box. $495 +$5 ship
$50 discount if prepaid by check, Visa, MC
[ ] **Compiler Option**: (PLUS) adds runtime compiler and Communication Package, with manuals packed in a Doc Box. $295+ $4 ship
$50 discount if prepaid by check, Visa, MC.
[ ] **Option**: Common COMAL Reference $16.95
[ ] **Option**: UniDump $45 (for laser printers)
[ ] **Option**: UniMatrix $145 (matrix package)
[ ] **Option**: Hercules Support $85
[ ] **Option**: Btrieve interface $25 /$110 multi
[ ] **Option**: XQL interface $110

db=Doc Box pages, requires a Doc Box for use.
***=subject to customs/ship variations/availability.**

BOOKS: (remember to add on the shipping fee)

[ ] $16.95 **Beginning COMAL¤**
#8 best seller by Borge Christensen
333 pages - General Textbook
Beginners text book, elementary school level, written by the founder of COMAL. This book is an easy reading text. You should find Borge has a good writing style, with a definite European flair.

[ ] $16.95 **Foundations with COMAL¤**
#4 best seller in Jan 88 by John Kelly
363 pages - General Textbook
Beginners text book, Jr/Sr High School level, including a section on C64 turtle graphics. A good text for those serious about programming. (only 4 copies left)

[ ] $17.95 **Introduction to COMAL 2.0¤**
by J William Leary - (SOLD OUT)
272 pages plus a 64 page answer book. A Jr/Sr High School level text book. Spiral bound. SORRY - ALL SOLD OUT! Watch for reprint news.

[ ] $3.95 **COMAL From A to Z¤**
#1 all time best seller by Borge Christensen
64 pages - Mini 0.14 Reference book
Written by the founder of COMAL

[ ] $3.95 **COMAL Workbook¤**
#1 best seller for Feb 88 by Gordon Shigley
69 pages - 0.14 Tutorial Workbook
Companion to the Tutorial Disk, great for beginners, full sized fill in the blank style.
[ ] **Tutorial Disk Option**: $9.95

[ ] $3.95 **Index to COMAL Today 1-12**
#9 best seller by Kevin Quiggle
52 page, 4,848 entry index to COMAL Today. The back issues of *COMAL Today* are a treasure trove of COMAL information and programs! This index is your key.
[ ] **Index Disk Option**: $9.95

[ ] $16.95 **Common COMAL Reference**^{db}
#3 best seller for Feb 88 by Len Lindsay
238 page detailed cross reference to the COMAL implementations in the USA (formerly *COMAL Cross Reference*)
Covers: C64 COMAL 2.0, C128 COMAL 2.0, CP/M COMAL 2.10, Mytech IBM PC COMAL 2.0, and UniComal IBM PC COMAL 2.1

[ ] $2.95 **C64 COMAL 2.0 Keywords**^{db}
#4 best seller for Feb 88 lists all the keywords built into the cartridge (including all 11 built-in packages) in alphabetic order complete with syntax and example.

[ ] $14.95 **CP/M COMAL Package Guide**^{db}
The guide to making your own packages for CP/M COMAL by Richard Bain
76 pages - advanced package reference

[ ] $14.95 **Library Functions/Procedures**^{db}
#1 best seller Dec 87 by Kevin Quiggle, 80 pages, over 100 0.14 procs/funcs, with disk

[ ] $4.95 **Cartridge Graphics & Sound¤**^{db}
#6 best seller by Captain COMAL
64 pages - 2.0 packages reference
Reference guide to all the commands in the 11 built in cartridge packages.

[ ] $14.95 **COMAL 2.0 Packages**^{db}
#7 all time best seller by Jesse Knight
108 pages with disk - package reference
How to write a package in Machine Code; includes C64 comsymb & supermon. For advanced users.

[ ] $14.95 **Package Library Vol 1**^{db}
compiled by David Stidolph
76 pages with disk - package collection
17 packages ready to use, many with source code, plus the Smooth Scroll Editor!

[ ] $14.95 **Package Library Vol 2**^{db}
67 pages with disk - package collection
24 example packages ready to use, most with source code, plus Disassembler, Re-Linker, De-Linker, Package Maker, Package Lister, and more. (includes windows)

[ ] $14.95 **COMAL Collage**^{db}
by Frank and Melody Tymon
168 pages with disk, 2.0 programming guide, including graphics and sprites tutorial with many full sized example programs.

[ ] $14.95 **3 Programs in Detail**^{db}
82 pages with disk by Doug Bittinger
Three 2.0 application programs explained: Blackbook (name/address system), Home Accountant, and BBS.

[ ] $14.95 **Graph Paper**^{db}
52 pages with disk by Garrett Hughes
Function graphing system for COMAL 2.0. The program can't be LISTed. Includes a version for the Commodore Mouse.

[ ] $14.95 **COMAL Quick / Utility 2 & 3**^{db}
#2 best seller Dec 87 by Jesse Knight
20 pages with 2 disks, fast loading COMAL 0.14, printer programs, utility programs.

db = Doc Box pages
¤ = while supplies last (out of print)

DISKS:

Disks are $9.95 each. Unless the label on the disk you receive specifically states that you may give out copies, our disks may not be copied or placed into club disk libraries. **Choose from the disks below:**

[ ] IBM Special Series Disk #1
[ ] CP/M COMAL Demo Disk ($5)
[ ] Mytech Amiga COMAL Demo Disk ($5)
[ ] Beginning COMAL disk §
[ ] Foundations with COMAL disk §
[ ] COMAL Handbook disk §
[ ] New: **Introduction to COMAL 2.0 disk** §
[ ] Today INDEX disk § (2 disks count as 1)
[ ] Games Disk #1 (0.14 & 2.0)
[ ] Modem Disk (0.14 & 2.0)
[ ] Article text files disk

Today Disks:

[ ] Today Disk (one disk type--circle choices):
1 2 3 4 5 20&21
[ ] Today Disk (double sided -- circle choices):
6 7 8 9 10 11 12 13 14 15 16 17 18 19 22

0.14 Disks:

[ ] Data Base Disk 0.14
[ ] New: Power Driver Tutorial Disk
[ ] Auto Run Demo Disk
[ ] Paradise Disk
[ ] Best of COMAL
[ ] Bricks Tutorial (2 disks count as 1)
[ ] Utility Disk 1
[ ] Slide Show disk (circle which): 1 2
[ ] Spanish COMAL
[ ] User Group 0.14 disks (circle numbers):
1 2 3 4 5 6 7 8 9 10 12

2.0 Disks:

[ ] Data Base Disk 2.0
[ ] Superchip Programs disk
[ ] Read & Run
[ ] Math & Science
[ ] Typing disk (2 disks count as 1)
[ ] Cart Demo (circle which): 1 2 3 4
[ ] 2.0 user disks (circle choices): 11 13 14 15

§ = these disks assume you have the book

Note: Some disks may be supplied on the back side of another disk. Disk format is Commodore 1541 unless specified otherwise. We replace any defective disk at no charge if you return the disk with a note explaining what is wrong with it. Some disks are being reduplicated and appropriately relabeled.

SPECIAL DISKS & DISK SETS:

[ ] $10.95 **Sprite Pak**
Two disk set. Huge collection of sprite images, sprite editors, viewers, and other sprite programs. For 0.14 and 2.0.

[ ] $12.95 **Font Pak**
Three disk set. Collection of many different character sets (fonts) for use with 0.14 and 2.0 including special font editors!

[ ] $14.95 **Graphics Pak**
Five disk set. Picture heaven. Includes Slide Show, Picture Compactor, Graphics Editor and lots of pictures (normal and compacted)

[ ] $32.95 **Sprite, Font & Graphics Pak**
All ten disks mentioned above!

[ ] $9.95 **C128 CP/M Graphics**
Graphics package on disk for use with CP/M COMAL on the C128. Includes turtle graphics and preliminary Font package.

[ ] $10.95 **Guitar Disks**
Three 0.14 disk set. Teaches guitar by playing songs while displaying the chords and words.

[ ] $14.95 **Cart Demo Disks**
Four disks full of programs demonstrating the many features of the C64 2.0 cartridge.

[ ] $10.95 **Shareware Disks**
Three disk set. Includes a full HazMat system (Hazardous Materials), an Expert System, Finger Print system, Traffic Calc, and a BBS program.

[ ] $14.95 **Superchip On Disk**
All the commands of Super Chip (but not the Auto Start feature) disk loaded.

[ ] $24.95 **Super Chip Source Code**
Full source code with minimal comments. Customize your own Super Chip. Add commands. Remove the ones you don't need.

[ ] $14.95 **2.0 User Group Disks**
Four disks set for the C64 COMAL cart.

[ ] $29.95 **0.14 User Group Disks**
Twelve disks (User Group disk 9 is a newsletter on disk system, double disk).

[ ] $29.95 **European Disk Set**
Twelve 2.0 disk set. Find out what COMALites in Europe are doing.

Banner Printer

by Paul Keck

Bannerprinter is... a banner printer. It can print the character images from ROM, a font in memory, or a font file (without having to link it!). Size can be full width of the paper to about an inch high.

```
USE font
USE char
DIM char'defn$ OF 8, line$ OF 120, filename$
OF 16
DIM font'choice$ OF 1, set'choice$ OF 1
set:=0
DIM e(0:8), fontdefn$ OF 2048
FOR i:=0 TO 7 DO e(i):=2^(7-i)
LOOP
  PAGE
  PRINT "Banner Printer"
  PRINT "=============="
  PRINT "This program prints out banners on "
  PRINT "continuous feed paper.  For the"
  PRINT "letter shapes, do you want to use"
  PRINT "ROM images, custom font, or a font"
  PRINT "file (r/c/f)?"
  REPEAT font'choice$:=KEY$ UNTIL font'c
  hoice$ IN "rcf" //wrap line
  PRINT ""13"Lowercase or uppercase/graphics "
  PRINT "(if using a font file, you probably"
  PRINT "want upper/graphics) (l/u)?"
  REPEAT set'choice$:=KEY$ UNTIL set'choice$
  IN "lu" //wrap line
  PRINT ""13"Enter height of each letter (1-10
  ):10", //wrap line
  INPUT AT 0,36,2: "": height
  PRINT "Enter width of each letter (1-10):5",
  IF height>10 THEN
    height:=10
  ELIF height<1 THEN
    height:=1
  ENDIF
  INPUT AT 0,35,2: "": width
  IF width>10 THEN
    width:=10
  ELIF width<1 THEN
    width:=1
  ENDIF
  CASE font'choice$ OF
  WHEN "c"
    set:=0
    IF set'choice$="l" THEN set:+1
  WHEN "r"
    set:=2
    IF set'choice$="l" THEN set:+1
  WHEN "f"
    PRINT ""13"Enter name of font file:";fi
    lename$, //wrap line
    INPUT AT 0,26,16: "": filename$
    PRINT ""13"Reading font definition..."
    OPEN FILE 2,filename$,READ
    fontdefn$:=GET$(2,2048)
    IF set'choice$="l" THEN fontdefn$:=G
    ET$(2,2048) //wrap line
    CLOSE FILE 2
  OTHERWISE
    NULL
  ENDCASE
  INPUT ""13"Enter string for banner: ": string$
  PRINT ""13"Printing..."13""
  FOR pos:=1 TO LEN(string$) DO
    PRINT string$(pos),
    SELECT OUTPUT "lp:"
    IF font'choice$="f" THEN
      char'defn$:=char'from'font$(screencode(
      string$(pos))) //wrap line
    ELSE
      getcharacter(set,screencode(string$(
      pos)),char'defn$) //wrap line
    ENDIF
    printchar(char'defn$,TRUE,height,width)
    SELECT OUTPUT "ds:"
  ENDFOR pos
  PRINT
ENDLOOP
//
PROC printchar(char'defn$,rotated,hgt,wdt)
  IF rotated THEN rotate(char'defn$)
  //PRINT ""5"", // boldface
  FOR i:=0 TO 7 DO
    row:=ORD(char'defn$(i+1))
    line$:=""
    FOR j:=0 TO 7 DO
```

more»

Banner Printer - continued

```
      IF row BITAND e(j) THEN
        pixel$:="*"
      ELSE
        pixel$:=" "
      ENDIF
      FOR vert:=1 TO hgt DO line$:+pixel$
    ENDFOR j
    FOR horiz:=1 TO wdt DO PRINT line$
  ENDFOR i
  //PRINT ""6"", // un-boldface (de-boldface?)
ENDPROC printchar
//
FUNC screencode(char$) CLOSED
  code:=ORD(char$)
  IF (code>=64 AND code<=95) OR (code>=128
  AND code<=191) THEN //wrap line
    code:-64
  ELIF code>=96 AND code<=127 THEN
    code:-32
  ELIF code>=192 AND code<=255 THEN
    code:-128
  ENDIF
  RETURN code
ENDFUNC screencode
//
FUNC char'from'font$(num)
  RETURN fontdefn$(num*8+1:num*8+9)
ENDFUNC char'from'font$
```
■

Allegan High --- Adventure Game

by Paul Keck

Allegan High is a simple adventure game written in 2.0 since I have no Power Driver. I believe that they will transfer without problem; however, it is probably pushing the memory limit for Power Driver (but it has a ton of comments in it that could be removed). The "*A.make*" type files which accompany the main program are datafile makers. "*A.make'all(2.0)*" is probably too long for Power Driver users, so the other two can be run separately. This creates the SEQ and REL files. I would be able to write an article about text adventure construction, if you would like. ■

CP/M Notes

by James Synnamon

I tried to learn about CP/M. Very frustrating. Until CP/M COMAL I didn't have much hope. I have put together a list of editing commands I have found. This may be useful to other C128 CP/M COMAL users:

«ctrl»-C Clear screen; cursor to top left corner
«ctrl»-D Same as gray cursor key forward
«ctrl»-E Same as gray cursor key up
«ctrl»-F Same as gray cursor key forward
«ctrl»-G Ring bell
«ctrl»-H Same as *«inst/del»*
«ctrl»-J Same as gray cursor key down
«ctrl»-K Erases characters to the end of line
«ctrl»-L Same as *«ctrl»*-C
«ctrl»-M Same as *«return»*
«ctrl»-P Cursor erases the line it is on which is replaced by the line below it, moving all lines below up one line. The cursor moves to the start of the line it is on
«ctrl»-Q The opposite of *«ctrl»*-P
«ctrl»-R Cursor to top left corner
«ctrl»-S Same as gray cursor left
«ctrl»-T Deletes character under cursor, rest of line slides over to left to fill the gap
«ctrl»-V Same as *«inst/del»*
«ctrl»-X Same as gray cursor down
«ctrl»-# (# is one of number keys at top of keyboard) change character colors
«ctrl»-# (# is one of number keys on keypad) change screen background color
«ctrl»-= Makes a tilde
«ctrl»-: Makes a {
«ctrl»-; Makes a }
«ctrl»-^ (the ^ is next to the *) makes a vertical line

«esc» Moves cursor to start of next line
«f3» DIR (does disk directory)
«f4» DIR (does disk directory)
«csr» The up/down cursor key next to the right hand shift key will toggle nearly 3 lines of characters. It is useful. ■

Christmas Song

by Gary Parkin

This program plays the favored Christmas song, Hark the Herald Angels Sing. Since it will soon be Christmas season, it seemed appropriate to publish this song.

```
// save "hark.xmas.song"
// by Gary Parkin
//
USE system
USE sound
DIM song$ OF 200
//
adsr(1,0,4,15,10)
adsr(2,0,9,2,9)
sync(2,1)
ringmod(2,1)
//
on:=TRUE; off:=FALSE
//
song$:="g4c5c5zzb4c5e5e5d5pp"
DATA 8,8,8,3,3,8,8,8,8
//
song$:+"g5g5g5zzf5e5d5e5zzxx"
DATA 8,8,9,3,3,8,8,8,10
//
song$:+"g4c5c5zzb4c5e5e5d5pp"
DATA 8,8,8,3,3,8,8,8,8
//
song$:+"g5d5d5zzc5b4a4g4zzxx"
DATA 8,8,8,3,3,8,8,8,10
//
song$:+"g5g5g5c5f5e5e5d5pp"
DATA 8,8,8,8,8,8,8,8
//
song$:+"g5g5g5c5f5e5e5d5xx"
DATA 8,8,8,8,8,8,8,8
//
song$:+"a5a5a5g5f5e5f5zzpp"
DATA 12,2,8,8,8,8,8,8
//
song$:+"d5e5f5g5c5c5d5e5zzpp"
DATA 8,4,4,12,4,8,8,8,8
//
song$:+"a5a5a5g5f5e5f5zzpp"
DATA 12,2,8,8,8,8,8,8
//
song$:+"d5e5f5g5xxc5c5d5c5pp"
DATA 9,5,5,14,8,12,12,16
//
//
ch:=1; g1:=off; g2:=on
FOR l:=1 TO LEN(song$)/2 DO
  code$:=song$(ch:ch+1)
  ch:+2
  IF code$="pp" THEN PRINT ""17""
  IF code$="xx" THEN
    IF g1=on THEN
      g1:=off
      PRINT ""17""
    ELIF g1=off THEN
      g1:=on
      PRINT ""17""
    ENDIF
  ENDIF
  //
  IF code$<>"pp" AND code$<>"xx" THEN
    READ wait
    PRINT code$;wait;
    play(code$)
  ENDIF
ENDFOR l
//
PROC play(code$)
  IF code$<>"zz" THEN
    note(1,code$)
    note(2,code$)
    gate(1,g1)
    gate(2,g2)
  ENDIF
  delay(wait*1.5)
  gate(1,0)
  gate(2,0)
ENDPROC play
//
PROC delay(sec'32)
  TIME 0
  WHILE TIME<1.875*sec'32 DO NULL
ENDPROC delay ■
```

Envelope Printer

by Gary Parkin

This is a short program that will print the address on an envelope. Adjust the print'envel procedure for your printer as needed.

```
// save "envelope program"
// by Gary Parkin
done:=FALSE
call'screen
REPEAT
  PAGE
  INPUT "Another Envelope? (y/n): ": yn$
  IF yn$="n" THEN
    done:=TRUE
  ELSE
    call'screen
  ENDIF
UNTIL done
END "Done"
//
PROC call'screen
  PAGE
  INPUT AT 6,4: "Insert envelope-hit return":r$
  PAGE
  PRINT "* * * *  envelope program  * * * *"
  PRINT AT 3,1: "Enter Name:"
  INPUT AT 5,1: name$
  PRINT AT 7,1: "Enter Address:"
  INPUT AT 9,1: address$
  PRINT AT 11,1: "Enter City:"
  INPUT AT 13,1: city$
  PRINT AT 15,1: "Enter State:"
  INPUT AT 17,1: state$
  INPUT AT 19,1: "Enter Zip: ": zip$
  INPUT AT 21,1: "Attention: ": attention$
  do'math
ENDPROC call'screen
//
PROC do'math
  n:=LEN(name$)
  a:=LEN(address$)
  c:=LEN(city$)
  s:=LEN(state$)
  z:=LEN(zip$)
  cs:=c+s+2
  env:=95 // full size envelope
  larger:=0; leftover:=0; zipover:=0
  compair(n,a)
  compair(larger,cs)
  zipover:=cs-z
  leftover:=(env-larger) DIV 2
  print'envel
ENDPROC do'math
//
PROC compair(first,last)
  IF first>last THEN larger:=first
  IF first<last THEN larger:=last
  IF first=last THEN larger:=first
ENDPROC compair
//
PROC print'envel
  SELECT OUTPUT "lp:" //IBM use "lpt1:"
  FOR l:=1 TO 7 DO PRINT
  PRINT TAB(leftover);name$
  PRINT TAB(leftover);address$
  PRINT TAB(leftover);city$+", "+state$
  PRINT TAB(leftover+zipover);zip$
  IF attention$>"" THEN
    PRINT
    PRINT TAB(leftover);"Attn:";attention$
  ENDIF
  PAGE
  SELECT OUTPUT "ds:" //IBM use "con:"
ENDPROC print'envel ■
```

New 2.0 Function --- DIGITS

Digits returns true if the string can be passed to VAL. This has some uses. Or you could use your own Value function (see page 37 and the Programming article in this issue).

```
0040 FUNC digits(a$) CLOSED
0050   TRAP
0060     x:=VAL(a$)
0065     x:=TRUE
0070   HANDLER
0080     x:=FALSE
0090   ENDTRAP
0100   RETURN x
0110 ENDFUNC digits ■
```

Diffusion Limited Aggregation

by Jim Frogge

This program simulates a process known as "*diffusion limited aggregation*". It yields a fractal pattern that is experimentally observed in several areas (such as the electrolytic deposition of various metals). My students and I have even been able to produce "*crystals*" of $CuSO_4$ - $5H_2O$ that closely match this pattern. The idea came from <u>Scientific American, Jan 1987, Fractal Growth by Leonard Sander</u>. This article describes how the fractal concept may be used as a modeling paradigm.

Along with my program is a typical example of the results of a 1000 point run (*dif'lim.hrg*). The idea: material is allowed to precipitate on a seed crystal in a nonequilibrium environment. Eight particles are visible at any given time. The initial stages of the process are rather slow, but middle and late phases proceed rapidly. The screen results are good, but I find the printed copy to be of more interest. If there is interest, I could present a two page article explaining the Math, Chemistry, and ideas for program modification.

[*This program creates a "seed" crystal in the center of the screen. Other particles move randomly until they either collide with the "seed" or move off the screen to the right.*]

```
USE math
USE graphics
USE system
USE sprites
TIME 0
constants'and'initial'values
create'sprites; seed'crystal
initial'release
REPEAT
  check'for'halt; why'stopped
UNTIL particle'count#=1000
//   new release constants are next
PROC constants'and'initial'values
  xlow:=20; xhigh:=40; ylow:=0; yhigh:=199
  mean'pathx:=60; errorx:=5; bias'x:=0
  mean'pathy:=60; errory:=5; bias'y:=0
  rangex:=mean'pathx+errorx //alter above
  rangey:=mean'pathy+errory //for path
  flag:=-1; speed:=5; particle'count#:=0
ENDPROC constants'and'initial'values
//
PROC create'sprites
  background(0); fullscreen; border(-1)
  pencolor(7)
  DIM drawing$ OF 64
  drawing$:=""224""+""0""+""0""+""224""+""0""+
  ""0""+""224""//wrap line
  FOR i#:=8 TO 64 DO drawing$:+""0""
  define(1,drawing$)
  FOR i#:=0 TO 7 DO
    identify(i#,1); showsprite(i#)
    spritecolor(i#,i#+1)
  ENDFOR i#
ENDPROC create'sprites
//
PROC seed'crystal
  plot(159,101); plot(160,101); plot(161,101)
  plot(159,100); plot(160,100); plot(161,100)
  plot(159,99); plot(160,99); plot(161,99)
ENDPROC seed'crystal
PROC initial'release
  FOR i#:=0 TO 7 DO release'new'particle(i#)
ENDPROC initial'release
//
PROC check'for'halt
  LOOP
    FOR i#:=0 TO 7 DO
      IF NOT moving(i#) THEN flag:=i#
    ENDFOR i#
    EXIT WHEN flag>-1
  ENDLOOP
ENDPROC check'for'halt
//
PROC why'stopped
  IF spriteinq(flag,11) THEN//sprit/data colsn
    plot(spritex(flag)+1,spritey(flag)-1)
    bell(1) //position is upper left of sprite
    release'new'particle(flag) //collision
  ELIF spritex(flag)>319 THEN
    release'new'particle(flag) //off'screen
```

more»

```
  ELSE
    get'new'move(flag) //end of path
  ENDIF // next reset flag after finding out
  flag:=-1 // cause of stop (only spot)
ENDPROC why'stopped
//
PROC release'new'particle(spriteno)
  xc:=RND(xlow,xhigh); yc:=RND(ylow,yhigh)
  spritepos(spriteno,xc,yc)
  get'new'move(spriteno); particle'count#:+1
ENDPROC release'new'particle
PROC get'new'move(spriteno)
  REPEAT
    newx:=spritex(spriteno)+RND((-rangex+
    bias'x),rangex+bias'x)//wrap line
    newy:=spritey(spriteno)+RND((-rangey+
    bias'y),rangey+bias'y)//wrap line
  UNTIL newx>0 AND THEN newy>0 AND TH
  EN newy<199 //wrap line-drift right
  movesprite(spriteno,newx,newy,step'(
  spriteno,newx,newy),%00000100)//wrap
ENDPROC get'new'move
FUNC step'(spriteno,x,y)
  RETURN speed*distance(spritex(spriteno),
  spritey(spriteno),x,y)+1//wrap line
ENDFUNC step' ■
```

Sample BOOT program for 2.0

This is a program one users runs first:

```
USE system
textcolors(6,6,1); quote'mode(0)
keywords'in'upper'case(0)
names'in'upper'case(0)
POKE $c850,6
POKE $c851,6
POKE $c852,1
defkey(1,"list "11"") //set up function keys
defkey(2,"auto "11"")
defkey(3,""2"run"11""13"")
defkey(4,"del "11"")
defkey(5,""11"pass""i0:"2"")
defkey(6,""11"select input""bat.cl"157""157"")
defkey(7,""11"dir""0:***"157""157""157"")
defkey(8," load""0:"11"") ■
```

Tower

by Luther Hux

The 3D airplane in *COMAL Today #19* was fun to create and a natural for someone like myself who flies radio controlled model airplanes. So what would be my next artwork challenge? I was hoping to select something that would not look like just another wire drawing. Looking over the many aerial photos I have taken using a remotely operated camera aboard an R/C model, I came across a photo of a tower at a theme park. The curves and cross hatch would be a real challenge but it would look great as a 3D wire drawing. After entering about 1,500 data items the tower was as finished as I cared to make it. A more complex cross hatch would be counter productive as it would make the tower nearly solid at a smaller scale. This is not intended to be the tower in Paris, just a theme park tower that looks somewhat like it. A pair of gate post were added to the ground detail to aid in seeing rotation.

Some combinations of larger input numbers may cause the program to draw a line in the *opposite direction of that specified in the data. Therefore, there are some limits suggested in* the input screen.

The rotation/perspective routine is identical to the airplane. Only the data has changed. However, since the airplane program was presented it has occurred to me that not everyone would have the article explaining the program on hand. Therefore an instruction screen has been added and an updated 3D airplane with a similar instruction screen is now also available. The need was made clear when a friend tried to answer the "scale" question with "1/2 scale". He then tried to answer "yaw, pitch, roll" with "yes".

Enjoy the view. The program is on *Today Disk #23*. ■

COMAL in the Classroom

by Carmen Sorvillo

For the past two years I've used COMAL 0.14 in the classroom at Bishop Loughlin High School in Brooklyn, New York. As Art Department Coordinator, I initiated a Computer Graphics course that would make use of the school's C64s. Since you're reading this in *COMAL Today*, I don't need to list the many reasons that made COMAL the language of choice. Suffice it to say that high school students take to COMAL very well and that COMAL's structure made grading program listings a snap.

The programming assignments ranged from creating simple displays of students' names printed in stars to animated sprites complete with turtle graphics backgrounds. They required the students to get graphic results using certain COMAL structures and/or graphics screen capabilities, but were open enough to allow for greatly varying levels of student creativity and achievement. The finished programs very often went beyond my initial expectations. Although an experienced programmer could in many cases work out shorter routines to accomplish the same ends, the programs accompanying this article represent the joy of discovery that COMAL can offer beginners within the time constraints of the classroom.

Dribble (by Floyd, period 5) is a computer animation that uses the Commodore graphic symbols to draw the pictures. A REPEAT loop is used in the main program to move the dribbler across the screen with incremented TAB values (on *Today Disk #23* only).

High'card (by John, period 5) is a game of chance that utilizes random numbers, Commodore graphic symbols, color, and COMAL's REPEAT, FOR, IF, and CASE structures. This is a great beginner's program with lots of features that exceeded the assigned requirements (on *Today Disk #23* only).

Computer (by Brett, period 2) is a hi-res drawing. The plotting of X-Y coordinates was used to transfer the student's drawing from graph paper to the graphic screen. This program also makes use of procedures with parameters that allow rows of keys to be drawn at desired locations. Each key is drawn by a procedure with parameters that uses turtle graphics to make a rectangle of variable size.

Cowboy (by Felix, period 2) is an animated sprite program. It uses one sprite that switches back and forth between two different images of a rider on his horse. The background landscape was drawn on the COMAL multi-color graphic screen.

COMPUTER

```
// brett - period 2
init
model
bar
row'keys(90,50)
row'keys(90,60)
row'keys(90,70)
column'keys(70,50)
column'keys(230,50)
fill'it
//
PROC init
  BORDER 0
  BACKGROUND 1
  PENCOLOR 0
  SETGRAPHIC 0
  FULLSCREEN
  HIDETURTLE
ENDPROC init
//
PROC model
  MOVETO 70,90
  DRAWTO 70,190
  DRAWTO 250,190
  DRAWTO 250,90
  DRAWTO 30,20
```

more»

```
  DRAWTO 290,20
  DRAWTO 250,90
  MOVETO 230,100
  DRAWTO 90,100
  DRAWTO 90,180
  DRAWTO 230,180
  DRAWTO 230,100
ENDPROC model
//
PROC bar
  MOVETO 90,30
  keys(10,130)
ENDPROC bar
//
PROC row'keys(x,y)
  FOR count:=1 TO 10 DO
    MOVETO x,y
    keys(10,10)
    x:+13
  ENDFOR count
ENDPROC row'keys
//
PROC column'keys(x,y)
  FOR count:=1 TO 3 DO
    MOVETO x,y
    keys(10,20)
    y:+10
  ENDFOR count
ENDPROC column'keys
//
PROC keys(width,length)
  FOR sides:=1 TO 2 DO
    FORWARD width
    RIGHT 90
    FORWARD length
    RIGHT 90
  ENDFOR sides
ENDPROC keys
//
PROC fill'it
  FILL 100,110
ENDPROC fill'it
```

COWBOY

```
// felix - period 2
init
backdrop
fill'it
rider'sprite
rider'move
//
PROC init
  BORDER 0
  BACKGROUND 1
  PENCOLOR 0
  SETGRAPHIC 1
  HIDETURTLE
  FULLSCREEN
ENDPROC init
//
PROC rider'sprite
  DIM rider$ OF 64
  FOR info:=1 TO 64 DO
    READ info'data
    rider$(info):=CHR$(info'data)
  ENDFOR info
  DEFINE 1,rider$
  DATA 0,0,0,0,28,0,0,28,0
  DATA 6,28,0,15,12,0,125,156,0
  DATA 125,196,0,13,12,0,13,142,152
  DATA 15,255,242,15,237,242,7,237,242
  DATA 3,237,240,63,255,224,126,12,112
  DATA 96,12,48,96,28,24,96,0,48
  DATA 96,0,128,0,0,192,0,0,0
  DATA 0 // end of rider data
  //
  DIM rider2$ OF 64
  FOR info2:=1 TO 64 DO
    READ info2'data
    rider2$(info2):=CHR$(info2'data)
  ENDFOR info2
  DEFINE 2,rider2$
  DATA 0,0,0,0,28,0,0,28,0
  DATA 6,28,0,15,12,0,125,156,1
  DATA 125,196,2,13,12,4,13,142,248
  DATA 15,255,240,15,237,240,7,237,240
  DATA 3,237,240,7,255,252,12,12,62
  DATA 24,12,6,24,28,6,12,0,6
```

more»

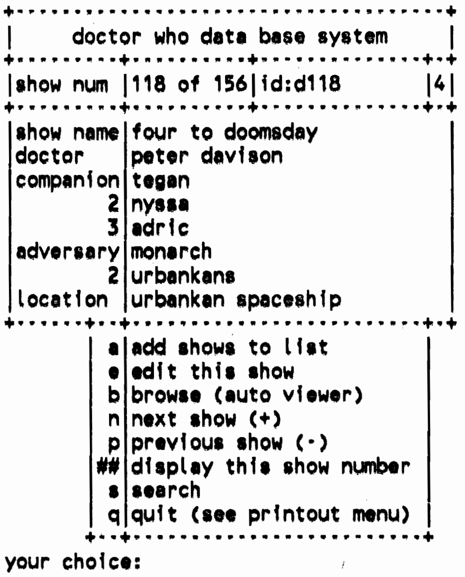
```
  DATA 6,0,6,3,0,0,0,0,0
  DATA 0,0,0,0,0,0,0,0,0
  DATA 0 // end of rider2 data
ENDPROC rider'sprite
//
PROC rider'move
  SPRITECOLOR 1,0
  SPRITESIZE 1,FALSE,FALSE
  PRIORITY 1,FALSE
  REPEAT
    FOR xpos:=310 TO 1 STEP -9 DO
      SPRITEPOS 1,xpos,100
      IDENTIFY 1,1
      FOR pause:=1 TO 100 DO NULL
      IDENTIFY 1,2
      FOR pause:=1 TO 100 DO NULL
    ENDFOR xpos
  UNTIL TRUE=FALSE
ENDPROC rider'move
//
PROC backdrop
  MOVETO 0,80
  DRAWTO 320,80
  MOVETO 20,80
  DRAWTO 20,90
  DRAWTO 15,90
  DRAWTO 15,100
  DRAWTO 20,100
  DRAWTO 20,96
  DRAWTO 21,96
  DRAWTO 21,105
  DRAWTO 26,111
  DRAWTO 30,105
  DRAWTO 30,80
  MOVETO 0,80
  DRAWTO 60,170
  DRAWTO 101,80
  MOVETO 92,100
  DRAWTO 110,129
  DRAWTO 136,80
  MOVETO 220,105
  DRAWTO 220,110
  DRAWTO 230,120
  DRAWTO 270,120
  DRAWTO 280,110
  DRAWTO 280,105
  DRAWTO 220,105
  MOVETO 225,80
  DRAWTO 225,105
  MOVETO 275,80
  DRAWTO 275,105
  MOVETO 256,120
  DRAWTO 256,130
  DRAWTO 263,130
  DRAWTO 263,120
  MOVETO 230,80
  DRAWTO 230,97
  DRAWTO 239,97
  DRAWTO 239,80
  MOVETO 260,94
  DRAWTO 260,100
  DRAWTO 270,100
  DRAWTO 270,94
  DRAWTO 260,94
  MOVETO 49,150
  DRAWTO 55,140
  DRAWTO 60,150
  DRAWTO 65,130
  DRAWTO 70,147
  MOVETO 100,110
  DRAWTO 105,103
  DRAWTO 110,110
  DRAWTO 115,99
  DRAWTO 120,109
ENDPROC backdrop
//
PROC fill'it
  PENCOLOR 9
  FILL 0,0
  FILL 110,90
  FILL 50,100
  FILL 10,85
  PENCOLOR 12
  FILL 50,100
  PENCOLOR 0
  FILL 260,125
  FILL 250,110
  PENCOLOR 5
  FILL 25,90
  FILL 250,90
ENDPROC fill'it ■
```

Programming: The Details

by Len Lindsay

For the past years, I have been so busy just keeping the COMAL Users Group running, that I didn't have time to actually use COMAL myself. So, I gave myself a project that should be similar to projects you may choose... take an existing COMAL program and extend it, modify it, adapt it... make it do more, make it more specific. Since I like Doctor Who shows (shown on many PBS stations), I decided to extend the Doctor Who Database system published in *COMAL Today 15*. Of course, many of the procedures and functions in this program can be adapted to other programming projects that you may decide to try. After this article are a few related quick programs, mostly dealing with random files.

I will try to explain many of the things that changed in this program. I may be a bit rusty on programming now, so that may show. It has been quite a while since we printed a program interspersed with explanatory text. And that is the way I will present this extended program.



```
+----------------------------------------+
|      doctor who data base system       |
+---------+----------+-----------------+-+
|show num |118 of 156|id:d118          |4|
+---------+----------+-----------------+-+
|show name|four to doomsday              |
|doctor   |peter davison                 |
|companion|tegan                         |
|        2|nyssa                         |
|        3|adric                         |
|adversary|monarch                       |
|        2|urbankans                     |
|location |urbankan spaceship            |
+------+--+----------------------------+-+
       | a|add shows to list           |
       | e|edit this show              |
       | b|browse (auto viewer)        |
       | n|next show (+)               |
       | p|previous show (-)           |
       |##|display this show number    |
       | s|search                      |
       | q|quit (see printout menu)    |
       +--+----------------------------+
your choice:
```

Since the procedures and functions can be entered in any order, before, after, or in between the main program, you may type them in the order you wish. I like to type the main program first, followed by the procedures. Pascal requires the opposite, and you can do it that way to if you like.

To get the most out of this, you should get out your copy of *COMAL Today 15* and re-read the article explaining the basic principles to the database system, starting on page 26. If you don't have a copy of issue 15, by all means get one now. We only have about 100 copies left!

So far, there have been 151 Doctor Who shows broadcast. This database keeps track of specific information about each of the shows, one screen per show. After publishing the original program, I wanted to show how easy it would be to adapt it to keep track of other shows. I chose Star Trek as an example, and *Today Disk 16* included the program: **Star Trek: The Database.**

At that point, I had two very similar large programs. One specifically for Doctor Who shows, and the other just for Star Trek shows. The next step was to make the program generic enough to handle both, and even more. This was not to hard to do. Before I explain how I did it, I will first review the structure of the random file used by the database system to store all the information.

A random file can hold many records. In this case, information about one show is one record. There are limits on how much information can be stored in one record in a random file. These limits are generally imposed by the computer system itself, rather than by COMAL. For example, IBM PC COMAL has a limit of 65535 for a record, while C64 COMAL can hold only 254 characters maximum per record.

more»

A random access file, allows quick access to any record, without having to read the ones before it first (as with a sequential file). However, every record written to disk is always the same length, regardless of the actual data being stored. You set a record length once for the file, and that cannot be easily changed later. If the record length is 200, it would take 200 bytes to store the number 5 as a record! Thus to conserve disk space, you should try to calculate the most characters you will ever need to write to a record in the file, and use that maximum as the record length for the file.

Calculating the number of characters you will need varies from system to system... and also depends on whether you use READ and WRITE FILE statements, or INPUT and PRINT FILE statements. The former set is more efficient, and more implementation specific, storing numbers in a binary fashion, and strings preceded by a length counter. It is what I use in this program. The latter has more limitations, but is more universal, as the data is written as ASCII separated by carriage return delimiters.

Here are some rules to follow when calculating how much room you need for a record:

- determine what will be stored in the record
- allow for a maximum for each variable stored
- allow for delimiters in ASCII files
  - IBM uses CHR$(13)+CHR$(10)
  - C64 uses CHR$(13)
- allow for string length counters in binary files
  - use the max length of the string plus 2
- allow for set numeric storage in binary files
  - IBM requires 4 bytes for each integer
  - IBM requires 8 bytes for each real
  - C64 2.0 requires 2 bytes for each integer
  - C64 0.14 requires 5 bytes for each integer
  - C64 requires 5 bytes for each real

Each record in the Dr Who database holds the data for one show. Here is how I calculate the total maximum length for the record:

```
29 - show name (27 + 2 for counter)
29 - doctor name (27 + 2 for counter)
29 - companion (27 + 2 for counter)
29 - companion (27 + 2 for counter)
29 - companion (27 + 2 for counter)
29 - adversary (27 + 2 for counter)
29 - adversary (27 + 2 for counter)
29 - location (27 + 2 for counter)
13 - id (11 + 2 for counter)
 3 - mark/episode character (1 + 2 for counter)
===
248 total maximum length
```

In various shows, there may be more than one companion or adversary, so I allowed for 3 companions, and two adversaries. In real use, it can store more than one name in a field, as long as it doesn't exceed the 27 character limit. There are 10 fields in each record. Each variable that is stored in the record is called a field. You need to calculate the maximum possible for each field, then add the length of all the fields to get the total for the whole record. The first 8 fields in my record are each 27 characters long (plus the two byte counter for each). They are exactly the same length because they are displayed on the screen in a box, one after another. Since they are the same length, I used a string array for them. You can see it dimensioned in <u>PROC dims</u>:

```
DIM field$(1:8) OF 27
```

Notice, that I only dimension for the length needed for the actual string. The extra two byte string counter needs to be allowed for only when writing the values to a file.

So, after all this, here is what the OPEN statement might look like for a random access file to hold the information we just discussed:

```
OPEN FILE 2,"filename",RANDOM 248
```

more»

The 248 is the record length for the file. It can also be a variable, as I used in the program:

```
record'length=248
filename$="docwho.ran"
OPEN FILE 2,filename$,RANDOM record'length
```

When using a random access file, remember that trying to access a record that doesn't exist causes an error (of course). So you must have a way to know how many records you have written. A common method is to use the first record to store the number of the last record. Then each time the program runs, it reads the first record, and that tells it the number of the last! For example, my original program had this simple procedure:

```
PROC read'last // preliminary
  OPEN FILE 2,filename$,RANDOM record'length
  READ FILE 2,1: last'show
  CLOSE FILE 2
ENDPROC read'last
```

Of course, as records are added, the number stored in that first record must be increased, so a companion procedure was used:

```
PROC write'last // preliminary
  OPEN FILE 2,filename$,RANDOM record'length
  WRITE FILE 2,1: last'show
  CLOSE FILE 2
ENDPROC write'last
```

Since I am OPENing and CLOSEing the file in several places, I decided to set up a procedure to do these simple tasks. It may or may not be efficient, but it seemed like a nice idea.

```
PROC open'it
  OPEN FILE 2,filename$,RANDOM record'length
ENDPROC open'it
//
PROC close'it
  CLOSE FILE 2
ENDPROC close'it
```

Some COMAL systems could have a problem with trying to close a file that is not open. If this could happen, use this routine instead:

```
PROC close'it
  TRAP
    CLOSE FILE 2
  HANDLER
    NULL
  ENDTRAP
ENDPROC close'it
```

Then change the other two procedures like this:

```
PROC read'last // preliminary
  open'it
  READ FILE 2,1: last'show
  close'it
ENDPROC read'last
//
PROC write'last // preliminary
  open'it
  WRITE FILE 2,1: last'show
  close'it
ENDPROC write'last
```

It may already have occurred to you that by writing only one number in the first record, that the rest of the characters reserved for it are wasted. But, I found a way to use this extra space while expanding the program. In order for it to be flexible, I needed a way to customize the prompts for each field. For instance, Star Trek shows would not include a Doctor Who! And rather than companions, there might be guest stars! So, I decided to set up an array of prompts for the ten fields in my show display box. Each prompt could be up to 9 characters and fit nicely in the box, so this is how it was dimensioned:

```
DIM prompt$(1:10) OF 9
```

Then I calculated how much of the record it would take to store these prompts:

more»

10 * 11 = 110 (remember the 2 byte counter)

I still had lots of room left in that first record. And it is a good thing too, because one of the features of this show database system is a search capability. If more than one field is used for the same thing, I need a way to search each of the applicable fields when looking for a match.

For example, if searching for a show with cybermen as an adversary, I must search each of the two adversary fields in each show record. Now that the fields are flexible, with variable prompts, I won't know for sure, which of the fields contain similar information. Before, I just matched up each of the companion fields with the companion search text, and both adversary fields with the adversary search text. Now I had to come up with a more general method.

Here is how a simple, one to one match procedure would look (one search field for each show field):

```
PROC match'record // preliminary only!
  matching=FALSE
  FOR temp=1 TO 8 DO //check each main field
    check(find'field$(temp),field$(temp))
  ENDFOR temp
ENDPROC match'record
//
PROC check(find'text$,text$) // preliminary
  IF find'text$>"" THEN
    IF find'text$ IN text$ THEN matching:=
    TRUE//all lower case//wrapline
  ENDIF
ENDPROC check
```

Here is how this simple version of search matching works. Find'field$ is a string array, just like field$, only it stores the text we are searching for, while field$ stores the actual show information. First we initialize our matching flag to FALSE (for not a match). Then we loop through each of the 8 main fields, calling procedure check to see if there is a match. The first thing check does, is see if there is any text in this field to search for (we don't want to do a search if there is nothing to search for). If we need to search, it checks if find'text$ can be found IN text$. If there is a match, matching is set to be TRUE.

This simple search method only works on a one to one field basis. So, I devised a method of searching more than one field in a show record for the search text, if several fields were used for the same purpose (such as 2 adversary fields). I created yet another string array! This array was set up to store which fields should be searched for each search field. It may be easier to explain this by using the program as an example.

The fields in the Dr Who database are set up this way:

Field 1 = show name
Field 2 = doctor name
Field 3 = companion
Field 4 = companion
Field 5 = companion
Field 6 = adversary
Field 7 = adversary
Field 8 = location

When starting a search, the user types in text to look for in any of the fields. Let's analyze what should happen.

If text is typed in the first find field, it will be a search on the show name. Only the first field of each show record can hold the show name, so we only have to search field 1.

If text is typed in the second find field, we will be searching for the doctors name. Since only one show field can store the doctors name, again, we only need to look at just that field.

more»

However, if find text is typed in the third field, this is one of three fields used to store companion info for each show. We will need to check for a match against not only field 3, but also field 4 and 5.

This also applies if find text is typed into field 4. We would have to check for a match to it with fields 3, 4, and 5.

Since I have extra room in the first record (after storing the number of the last show and each of the 10 prompts), I use the remaining space to store information about compatible fields (fields 3, 4, and 5 are compatible, as are fields 6 and 7 in my Doctor Who database).

I set up a string array named __match$__ to keep track of compatible fields. It is dimensioned like this:

```
DIM match$(1:8) OF 8
```

Since there are 8 main fields, I only need 8 characters maximum to specify which show fields are compatible with each of the 8 find fields.

Next I had to make sure there was enough room in the first record to store this added information. Here are my calculations for the first record:

  8 - last show number «IBM» (5 for C64)
 39 - system name (37 + 2 count bytes)
110 - field prompts
      (10 sets of 9, + 2 count bytes each)
 80 - compatible fields
      (8 sets of 8, + 2 count bytes each)
====
237 total bytes required for first record.
      (234 for C64)

It would fit, since I have 248 bytes available for each record, and it only requires 237. So, next I modified my __read'last__ and __write'last__ procedures to keep track of my prompts and compatible matching fields:

```
PROC write'last
  open'it
  WRITE FILE 2,1: last'show
  WRITE FILE 2: system'name$
  FOR temp:=1 TO 10 DO
    WRITE FILE 2: prompt$(temp)
  ENDFOR temp
  FOR temp:=1 TO 8 DO
    WRITE FILE 2: match$(temp)
  ENDFOR temp
  close'it
ENDPROC write'last
//
PROC read'last
  open'it
  READ FILE 2,1: last'show
  READ FILE 2: system'name$
  FOR temp:=1 TO 10 DO
    READ FILE 2: prompt$(temp)(1:9)
  ENDFOR temp
  FOR temp:=1 TO 8 DO
    READ FILE 2: match$(temp)
  ENDFOR temp
  close'it
ENDPROC read'last
```

I also had to modify my search routines so that they checked each compatible field based on the __match$__ values:

```
PROC match'record
  matching:=FALSE
  FOR temp:=1 TO 8 DO
    check(find'field$(temp),field$(temp))
    IF match$(temp)>"" THEN
      FOR cycle:=1 TO LEN(match$(temp)) DO
        this'field:=value(match$(temp)(
        cycle:cycle)) //wrap line
        check(find'field$(temp),field$(this'
        field)) //wrap line
      ENDFOR cycle
    ENDIF
  ENDFOR temp
```

more»

```
  check(find'id$,id$)
  check(find'mark$,mark$)
ENDPROC match'record
//
PROC check(find'text$,text$)
  IF find'text$>"" THEN
    IF find'text$ IN lower$(text$) THEN match
    ing:=TRUE//all lower case//wrapline
  ENDIF
ENDPROC check
```

You can see how the check procedure now checks each compatible field using the match$ array. It also now allows us to check for matching ID and MARK (now used for episode count). Notice that I use value instead of VAL to get a number from a string. In IBM COMAL and C64 COMAL 2.0, trying to take the VAL of a non-numeric string is an error. So, rather than chance crashing the program, I put together my own value routine, that returns 0 for any non-numeric string:

```
FUNC value(text$)
  TRAP
    RETURN VAL(text$)
  HANDLER
    RETURN 0
  ENDTRAP
ENDFUNC value
```

(In C64 Power Driver, the VAL does not crash on non-numerics, it just returns 0, as this routine here does).

While I was modifying things, I also solved the problem of UPPER or lower case mismatches that the original program had. If you asked to search for CYB in the original program, it would not match Cyb. Originally, to get around this limit, all characters had to be typed unshifted. It was embarrassing to have to tell people that they couldn't capitalize a persons name! It was fairly simple to solve that problem. I just convert the find text to all lower case ... then, as searches are made, I convert the show info into all lower case each time a field is checked. It was simple with IBM PC COMAL (which has more power and flexibility than C64 COMAL).

Here is the routine I use to convert text so that any UPPER case letters are changed to lowercase. First I check that there is some text to be converted. If not, I return the null string. Next, I loop through, checking each letter one at a time. I use pos to store which letter it is (or 0 for not an upper case letter). Then if pos is greater than 0, I replace that letter with its lower case equivalent... taking a substring of a text constant (the C64 COMALs can't do this, so you need to assign the lower case alphabet to a variable first, then use a substring of that variable... see the second listing).

```
FUNC lower$(text$) // <<< IBM ONLY >>>
  IF LEN(text$)<1 THEN RETURN ""
  FOR x:=1 TO LEN(text$) DO
    pos:=text$(x:x) IN "ABCDEFGHIJKLMNOP
    QRSTUVWXYZ" //wrap line
    IF pos THEN text$(x:x):="abcdefghijklmnop
    qrstuvwxyz"(pos:pos) //ibm only//wrap line
  ENDFOR x
  RETURN text$
ENDFUNC lower$
```

While the lower$ listed above only works with IBM COMAL, the listing below will work with all COMAL systems. However, at the start of the program, you need to add these lines:

```
DIM alphabet$ OF 26
alphabet$="abcdefghijklmnopqrstuvwxyz"

FUNC lower$(text$) // all COMAL systems
  IF LEN(text$)<1 THEN RETURN ""
  FOR x:=1 TO LEN(text$) DO
    pos:=text$(x:x) IN "ABCDEFGHIJKLMNOP
    QRSTUVWXYZ" //wrap line
    IF pos THEN text$(x:x):=alphabet$(pos:pos)
  ENDFOR x
  RETURN text$
ENDFUNC lower$
```

more»

It is important to remember that your database information is on the disk. It is not a good idea to leave such an important file open while not actually being directly used. If the power goes out, or something, you don't want the file to be open, especially with Commodore disks. So, I have two ways of accessing the records in the file.

If I will be reading record after record (like for a search), I don't close the file until I read the last record needed at that time. However, when editing the file, I want to read a record, and then close the file. The file stays closed until ready to write the record back again. At that time, the file is opened, the record written, and then closed again.

I only need one write’record routine, as I always will close the file after writing a record. With Commodore disk drives, this solves a disk drive bug. And it is safest, because it keeps the file closed when not being accessed. There is nothing in this entire large program that writes two or more records immediately one after another, so there is no need to keep the file open after a write.

```
FUNC write'record(show'number)
  IF show'number<1 THEN RETURN FALSE
  // ^^^ invalid record ^^^
  // convert show num into record num
  record'number:=show'number+1
  PRINT AT 24,1: "writing record...",SPC$(22),
  TRAP
    open'it
    WRITE FILE 2,record'number: field$(1)
    FOR temp:=2 TO 8 DO
      WRITE FILE 2: field$(temp)
    ENDFOR temp
    WRITE FILE 2: id$
    WRITE FILE 2: mark$
    close'it
    RETURN TRUE
  HANDLER
    PRINT AT 24,1: ("===> "+ERRTEXT$+
    SPC$(39))(1:39), //wrap line
    MOUNT
    close'it
    INPUT AT 25,1,0: "record not written...hit
    <return> :": reply$; //wrap line
    RETURN FALSE
  ENDTRAP
ENDFUNC write'record
```

Notice that this is a FUNCtion, not a procedure! It shows another way that functions can be used, rather than just for calculations. In this case, it returns either TRUE or FALSE, depending on whether or not the record was properly written. Any problems writing the record, and it returns FALSE. The program section that tried to write the record could deal with any errors.

Next, the procedure that deals with disk access without opening or closing the file:

```
PROC read'record(show'number)
  IF show'number<1 OR show'number>last'show T
  HEN RETURN //invalid record//wrap line
  //convert show num into record num
  record'number:=show'number+1
  READ FILE 2,record'number: field$(1)(1:27)
  FOR temp:=2 TO 8 DO
    READ FILE 2: field$(temp)(1:27)
  ENDFOR temp
  READ FILE 2: id$
  READ FILE 2: mark$
ENDPROC read'record
```

Now, to read one record only, it takes only a three line procedure:

```
PROC read'it(rec'num)
  open'it
  read'record(rec'num)
  close'it
ENDPROC read'it
```

more»

To read many records, one after another, just requires a loop:

```
FOR x=1 TO last'show DO read'record(x)
```

Of course, in the program, something must be done with the information once it is read, such as compare or display it.

The main program itself is still quite short. It is modified from the original a bit, especially in adding the printout request.

```
start'up
dim'years
REPEAT
  format'screen
  display'
  choices
UNTIL done
IF filename$="docwho.ran" THEN
  REPEAT
    ask'printout
  UNTIL done'printing
ENDIF
halt
```

Format'screen, display', and choices are basically the same as in the original, however, I removed the option to highlight (reverse field) certain shows on the screen display. This originally was based on whether or not there was anything stored in mark$. Now, every show has something stored there, as it is used to store the episode count for each show. These routines are listed at the end of this article.

I have been using the Doctor Who database system for the past 2 years, and found a few other things I missed in it. So, I added them! First I thought it would be nice to keep track of the number of episodes in each show. I decided to use the one character mark$ field for this purpose, since nearly every one of the shows had 9 or less episodes and only needed one character. I just used A for 10 and C for 12 (as in HEX) for the two long shows. On any reports, charts, or printouts, the A and C can be converted to 10 and 12 respectively.

Talk about printouts! That was one thing I really missed! So I wrote a quick printout routine. It just printed all the info for each show in tiny letters across one line on 14 inch wide paper. No page breaks or anything fancy. This was great to check my information with. But to make it a little more presentable, I later added nice page breaks with headers and page numbers. Here is what printout looks like (along with its companion procs newpage and header).

```
PROC printout
  count:=0 // line count per page
  pagenumber:=0 // page number for printout
  PAGE
  SELECT OUTPUT printer$
  INPUT "Initialize Epson small print? ":
  reply$ //wrap line
  IF reply$="Y" OR reply$="y" THEN
    PRINT CHR$(27)+"!D",//small print on epson
  ENDIF
  header
  FOR x:=1 TO last'show DO
    read'it(x)
    PRINT USING "###->": x;
    FOR y:=1 TO 8 DO
      IF y=1 THEN
        IF field$(y)(25:27)="   " THEN field$(
        y)(25:27):="("+mark$+")" //wrap line
      ENDIF
      PRINT field$(y);
    ENDFOR y
    PRINT // carriage return
    count:+1
    PRINT // blank line
    count:+1
    IF count>55 THEN newpage
  ENDFOR x
  PAGE
  SELECT OUTPUT screen$
ENDPROC printout
//
```

more»

```
PROC newpage
  PAGE
  count:=0
  header
ENDPROC newpage
//
PROC header
  pagenumber:+1
  PRINT "Doctor Who Shows - The List -",
  PRINT " Please advise of any corrections",
  PRINT " required",SPC$(41),"Page";pagenumber
  PRINT "Not for publication-copyright 1988",
  PRINT " Len Lindsay, 5501 Groveland Ter,",
  PRINT " Madison, WI 53716"
  PRINT //blank line
  count:+3
  PRINT SPC$(5);
  PRINT "show name        (episodes)";
  PRINT "doctor name                ";
  PRINT "companion                  ";
  PRINT "companion                  ";
  PRINT "companion                  ";
  PRINT "adversary          (other)";
  PRINT "adversary          (other)";
  PRINT "location & time            "
  count:+1
  PRINT SPC$(5);
  FOR x:=1 TO 8 DO PRINT "===============
  ============"; //wrap line
  PRINT //carriage return
  count:+2
ENDPROC header
```

Notice that at the beginning of **printout** I select the printer with a variable for the printer specification, rather than a string constant as usual. This lets me set the printer "location" once in the beginning of the program, and use the same one in all printing routines. For C64, it may not seem like much, since the printer is nearly always "lp:". But with IBM, it is significant, since there can be 3 printers hooked up at once (actually 5 if you also use the RS-232 ports), even more with special plug in boards. Just include one of these lines at the start of the program:

```
printer$:="lpt1:"; screen$:="con:" //<<<ibm
printer$="lp:";screen$="ds:"//<<<c64
```

Also note that I select the printer first, and then ask an INPUT question! The prompt for the INPUT statement still goes on the screen, even though I have selected the printer as the output location. COMAL takes care of that for you. I did it this way so that if a printer needed initialization, the special codes could be sent to the printer without having to select the printer specially. The initialization I have included is for the Epson printers. You may substitute the correct codes for your printer ... if it can print on the wide 14 inch paper.

Near the end of **printout** you can see how I check for page breaks. I check if more than 55 lines have been printed, and if so, call the **newpage** routine.

Newpage is a simple procedure. It just issues a form feed to the printer (PAGE), resets the line count to 0, and then calls **header**.

Header increments the page count, prints two lines across the top of the page followed by a blank line, and increments the line count by 3 (for those three lines). Next it prints column titles for each field and then uses ===== to "underline" each category. Of course, the line count is incremented by 2 for those two lines.

Another interesting piece of information about each show is the year it was first broadcast. I quickly added the array **year** to keep track of it, and wrote a procedure to assign a year to each show. I did this with FOR loops directly. It also could have been done with DATA statements.

```
PROC dim'years
  IF filename$<>"docwho.ran" THEN RETURN
  DIM year(0:last'show)
  FOR x:=0 TO 2 DO year(x):=1963
  FOR x:=3 TO 10 DO year(x):=1964
```

more»

```
  FOR x:=11 TO 21 DO year(x):=1965
  FOR x:=22 TO 31 DO year(x):=1966
  FOR x:=32 TO 40 DO year(x):=1967
  FOR x:=41 TO 47 DO year(x):=1968
  FOR x:=48 TO 50 DO year(x):=1969
  FOR x:=51 TO 54 DO year(x):=1970
  FOR x:=55 TO 59 DO year(x):=1971
  FOR x:=60 TO 65 DO year(x):=1972
  FOR x:=66 TO 70 DO year(x):=1973
  FOR x:=71 TO 75 DO year(x):=1974
  FOR x:=76 TO 83 DO year(x):=1975
  FOR x:=84 TO 88 DO year(x):=1976
  FOR x:=89 TO 95 DO year(x):=1977
  FOR x:=96 TO 102 DO year(x):=1978
  FOR x:=103 TO 108 DO year(x):=1979
  FOR x:=109 TO 113 DO year(x):=1980
  FOR x:=114 TO 116 DO year(x):=1981
  FOR x:=117 TO 123 DO year(x):=1982
  FOR x:=124 TO 130 DO year(x):=1983
  FOR x:=131 TO 138 DO year(x):=1984
  FOR x:=139 TO 143 DO year(x):=1985
  FOR x:=144 TO 147 DO year(x):=1986
  FOR x:=148 TO 151 DO year(x):=1987
  FOR x:=152 TO last'show DO year(x):=1988
ENDPROC dim'years
```

Since I am doing it directly in the program for now, I had to add one check line at the start, to make sure this was the Doctor Who shows currently being used (not Star Trek or something else). Later I hope to store this information in the database record itself, deleting this procedure entirely, and avoiding the possible conflict between various databases. To allow me to add this in later without having to rewrite the whole random file, I used a slightly larger number as my record length... **254 rather than 248:**

```
record'length=254
```

Next, I wanted to add a routine that made use of my HP LaserJet. I realize that most of you don't have Laser printers, but I do want to include it for those who do, and to show how easy the LaserJet can be controlled from COMAL. It requires lots and lots of messy control code sequences to do anything special. Here are some procedures I set up to send the proper codes to the LaserJet. Now, I can switch to **Helvetica** by name!

```
PROC manual'feed'laserjet
  PRINT CHR$(27)+"&l2H",
ENDPROC manual'feed'laserjet
//
PROC normal'feed'laserjet
  PRINT CHR$(27)+"&l1H",
ENDPROC normal'feed'laserjet
//
PROC helvetica
  PRINT CHR$(27)+"&l0O"+CHR$(27)+"(0U"+
  CHR$(27)+"(s1p14.4v0s1b4T", //wrap line
ENDPROC helvetica
//
PROC roman'bold
  PRINT CHR$(27)+"&100"+CHR$(27)+"(0U"+
  CHR$(27)+"(s1p10v0s1b5T", //wrap line
ENDPROC roman'bold
//
PROC italic
  PRINT CHR$(27)+"&100"+CHR$(27)+"(0U"+
  CHR$(27)+"(s1p10v1s0b5T", //wrap line
ENDPROC italic
//
PROC roman
  PRINT CHR$(27)+"&100"+CHR$(27)+"(0U"+
  CHR$(27)+"(s1p10v0s0b5T", //wrap line
ENDPROC roman
//
PROC reset'laserjet
  PRINT CHR$(27)+"E", // reset laserjet
ENDPROC reset'laserjet
//
PROC lineprinter
  PRINT CHR$(27)+"(8U"+CHR$(27)+
  "(s0p16.66h8.5v0s-1b0T", //wrap line
ENDPROC lineprinter
//
PROC lpi8
  PRINT CHR$(27)+"&l8D",//laserjet
ENDPROC lpi8
```

more»

```
//
PROC lpi6
  PRINT CHR$(27)+"&l6D", //laserjet
ENDPROC lpi6
//
PROC tiny'roman
  PRINT CHR$(27)+"&10O"+CHR$(27)+"(0U"+
  CHR$(27)+"(s1p8.0v0s-1b5T", //wrap line
ENDPROC tiny'roman
```

With my LaserJet, I could not use the 14 inch wide paper, so I needed to rethink how to printout all the information on the shows. It would have to be multiple lines per show, but I needed a clear way to do it. I decided that putting each of the three companion fields info in one column, and the two adversary fields into another column would do it. It just fit using lineprinter font (see the example chart page). Next I realized that the Epson could print this chart too ... without needing the wide paper. Now <u>this</u> would be handy for many of you! So I added a quick check at the beginning of the routine to initialize either Epson or LaserJet printers. Notice that this time, I don't select the printer until just before the INPUT line. The previous "menu" choices are PRINT statements and must show up on the screen.

```
PROC smallprint
  count:=0; pagenumber:=0
  prev'doctor$(1:27):="william hartnell"
  whoyear:=1963
  PAGE
  PRINT "Printer initialization required:"
  PRINT
  PRINT "E - Epson"
  PRINT "L - LaserJet (lineprinter font)"
  PRINT
  PRINT "N - None: don't change current
   printer  setting" //wrap line
  PRINT
  SELECT OUTPUT printer$
  INPUT "Your choice: ": reply$
  IF reply$="E" OR reply$="e" THEN
    PRINT CHR$(27)+"!D", // epson small print
    PRINT CHR$(27)+"0", // epson 8lpi
  ELIF reply$="L" OR reply$="l" THEN
    reset'laserjet
    lineprinter
    lpi8
  ENDIF
  read'it(1)
  small'header
  FOR x:=1 TO last'show DO
    pagetop:=FALSE
    read'it(x)
    whoyear:=year(x)
    check'page
    IF pagetop=TRUE THEN
      page'bottom
      PAGE
      count:=0
      small'header
      dividing'line
    ELIF field$(2)<>prev'doctor$ OR
    whoyear<>year(x-1) THEN //wrap line
      doc'header
      bottom'line
    ELSE
      dividing'line
    ENDIF
    small'line
    prev'doctor$:=field$(2)
  ENDFOR x
  page'bottom
  PAGE
  SELECT OUTPUT screen$
ENDPROC smallprint
```

Calculating proper page breaks is more difficult this time, due to multiple lines for one show. We don't want to have a page break in the middle of a show. Also, I did get a little fancy and have breaks each time the year changes (like from 1963 to 1964), as well as whenever the doctor changes (like from William Hartnell, to Patrick Troughton).

Next are the routines needed by the <u>smallprint</u> procedure. Note that I did not want to have a separate column just for the number of

more»

episodes, so I combined it with the show name (which usually had enough room). The few times it did not fit, I printed it on the next line below the show name. Also, note the variable secret is used to flag when the chart is for me. This shows how you can have a hidden feature (explained later). This chart also marks with a * the "missing" shows (also explained later).

```
PROC small'line
  IF field$(1)(25:27)="   " THEN
    eps'done:=TRUE
    field$(1)(25:27):="("+mark$+")"
  ELSE
    eps'done:=FALSE
  ENDIF
  lstart$:="| "
  IF (secret=TRUE AND id$>"") AND THEN id$(
  1:1)<>" " THEN lstart$:="|." //wrap line
  IF (secret=TRUE AND id$>"") AND THEN id$(
  1:1)="d" THEN lstart$:="|:" //wrap line
  PRINT lstart$,
  PRINT USING "###": x,
  IF missing(x) THEN
    PRINT "*",
  ELSE
    PRINT " ",
  ENDIF
  PRINT "|";capitalize$(field$(1));"|";
  PRINT capitalize$(field$(3));
  PRINT "|";capitalize$(field$(6));"|";
  PRINT capitalize$(field$(8));"|"
  count:+1
  IF field$(4)<>SPC$(27) OR field$(7)<>SPC$(
  27) THEN //wrap line
    episode$:="   "
    IF eps'done=FALSE THEN episode$:=
    "("+mark$+")" //wrap line
    PRINT "|     |";SPC$(24)+episode$;"|";
    PRINT capitalize$(field$(4));"|";
    PRINT capitalize$(field$(7));"|";SPC$(
    27);"|" //wrap line
    count:+1
  ENDIF
  IF field$(5)<>SPC$(27) THEN
    PRINT "|     |";SPC$(27);"|";
    PRINT capitalize$(field$(5));"|";
    PRINT SPC$(27);"|";SPC$(27);"|"
    count:+1
  ENDIF
ENDPROC small'line
//
PROC dividing'line
  PRINT "|-----|",29*"-","|",29*"-","|"
  ,29*"-","|",29*"-","|" //wrap line
  count:+1 //^^^ibm only, 29 dashes
ENDPROC dividing'line
//
PROC bottom'line
  PRINT "|"+125*"-"+"|" //ibm only, 125 dashes
  count:+1
ENDPROC bottom'line
```

This small'line routine is fairly smart. It doesn't do multi-lines for shows with no information in the extra fields (that is why I check if the whole field is all spaces (27 spaces)). You also might notice capitalize$. This is my way to make all lower case information look nicer. The capitalize function will capitalize the first letter in most words, but never two letter words (like in, on, of) nor the words the or and ... unless it is the first word in the field. I'm sure there are many ways of doing this, but I like this way (it doesn't affect the original value in the field either):

```
FUNC capitalize$(text$)
  IF text$="" THEN RETURN ""
  text$(1:1):=cap$(text$(1:1))
  FOR x:=2 TO LEN(text$)-2 DO
    prev'char$:=text$(x-1:x-1)
    IF prev'char$ IN " (,;-/&[." THEN
      IF text$(x+2:x+2)=" " THEN
        NULL // dont cap 2 letter words
      ELIF text$(x:x+2)="the" OR text$(
      x:x+2)="and" THEN //wrap line
        IF text$(x+3:x+3)<>" " THEN text$(
        x:x):=cap$(text$(x:x)) //wrap line
      ELSE
        text$(x:x):=cap$(text$(x:x))
      ENDIF
```

more»

```
    ENDIF
  ENDFOR x
  RETURN text$
ENDFUNC capitalize$
//
FUNC cap$(letter$)
  IF letter$="" THEN RETURN ""
  char:=letter$ IN "abcdefghijklmnopqrstuv
  wxyz" //wrap line
  IF char THEN
    RETURN "ABCDEFGHIJKLMNOPQRSTUVW
    XYZ"(char:char) //ibm only//wrap line
  ELSE
    RETURN letter$
  ENDIF
ENDFUNC cap$
```

Next is the routine that prints the header at the top of each page. Notice at the beginning I need to print 127 dashes across the page. IBM COMAL has a quick way to do it ... multiplied strings! Yes, I can say PRINT 127*"-". Other COMALs can use a FOR loop to print the dashes (I like many of the nice deluxe features in the IBM COMAL).

```
PROC small'header
  pagenumber:+1
  PRINT 127*"-" //ibm only, 127 dashes
  count:+1
  PRINT "| Doctor Who Digest --- Please advise
  of any corrections needed", //wrap line
  PRINT " --- Written permission required for
  publication ---",TAB(118);"Page";//wrap line
  PRINT USING "##": pagenumber;
  PRINT "|"
  count:+1
  doc'header
  title'header
ENDPROC small'header
```

As you can see, <u>small'header</u> in turn, calls on <u>doc'header</u> to print the header for a new doctor break, and <u>title'header</u> to print the titles of the columns once at the start of each page.

```
PROC doc'header
  bottom'line
  PRINT "|";capitalize$(trim$(field$(2)));
  PRINT TAB(64);whoyear;
  trim'doctor$:=trim$(field$(2))
  PRINT TAB(125-LEN(trim'doctor$));
  PRINT capitalize$(trim'doctor$);"|"
  count:+1
ENDPROC doc'header
//
PROC title'header
  bottom'line
  PRINT "| Show| Show Name      (Episodes)";
  PRINT "| Companions              |";
  PRINT "Adversaries       (Other) |";
  PRINT "Location & Time Period    |"
  count:+1
ENDPROC title'header
```

Checking for when to switch to a new page is a bit more complex here. We need to check if a doctor header will be needed or not, as well as how many lines the next show needs.

```
PROC check'page
  lines'needed:=2+2 // the line & footer
  IF field$(4)<>SPC$(27) OR field$(7)<>SPC$(
  27) THEN lines'needed:+1 //wrap line
  IF field$(5)<>SPC$(27) THEN lines'needed:+1
  IF field$(2)<>prev'doctor$ OR year(x)<>year(
  x-1) THEN lines'needed:+2 //wrap line
  IF count+lines'needed>76 THEN pagetop:=TRUE
ENDPROC check'page
```

Finally, the routines to print this chart is concluded by a procedure to print a footer on each page. You can add footers to your reports to print your name and address and such information as required, and not clutter up the header with it.

```
PROC page'bottom
  bottom'line
  PRINT "|";"* means all or part of the show",
  PRINT " is missing -- Copyright 1988 Len",
  PRINT " Lindsay, 5501 Groveland ,Madison,",
```

more»

```
  PRINT " WI 53716 --",TAB(118);"Page";
  PRINT USING "##": pagenumber;
  PRINT "|"
  PRINT 127*"-" //ibm only, 127 dashes
ENDPROC page'bottom
```

Remember, the variable secret is used in the small'line procedure to flag when the chart is for me. This shows how you can have a hidden feature. At the beginning of the program, it asks you what database you wish to use. You can expand this routine to include your special show databases. Meanwhile, it asks for you to reply D for Doctor Who, or S for Star Trek. However, if I reply L (for Len) then it sets up for Doctor Who, and also sets the secret flag to TRUE (you can use this secret trick in your programs too, perhaps use the first letter of your name as the secret letter). In this program, I only use this to add a period or colon before the show number if it has information in the ID field (this means that I have it on tape). You can use this same secret feature if you want now ... I told you about it.

```
PROC set'filename
  PRINT
  PRINT "D - Doctor Who"
  PRINT "S - Star Trek"
  PRINT
  INPUT "(D) (S) or filename: ": filename$
  secret:=FALSE //special
  CASE filename$ OF
  WHEN "D","d","W","w","l"
    IF filename$="l" THEN secret:=TRUE //secret
    filename$:="docwho.ran"
  WHEN "S","s","T","t"
    filename$:="startrek.ran"
  OTHERWISE
    NULL
  ENDCASE
ENDPROC set'filename
```

The Doctor Who shows date back 25 years, and quite a few of the old ones were destroyed in the 1970's. This is unfortunate. I thought these should be marked (by a *) on the chart, so that you will know why they are skipped by your PBS station! While a short little routine with data statements could have been used to identify each missing show, I chose a more elaborate method, mainly so I could look at the program and see where each show was marked as missing. Each year, it seems one of the missing shows turns up (in Australia or such), and the previously missing show can be released again (such as with Time Meddler and War Machines).

```
PROC dim'missing
  DIM missing(1:last'show)
  FOR x:=1 TO last'show DO missing(x):=FALSE
  missing(4):=TRUE // marco polo
  missing(8):=TRUE // reign of terror
  missing(14):=TRUE // crusades
  missing(18):=TRUE // galaxy four
  missing(19):=TRUE // mission to the unknown
  missing(20):=TRUE // myth makers
  missing(21):=TRUE // dalek masterplan
  missing(22):=TRUE // massacre
  missing(24):=TRUE // celestial toymaker
  missing(26):=TRUE // savages
  missing(28):=TRUE // smugglers
  missing(30):=TRUE // power of the daleks
  missing(31):=TRUE // highlanders
  missing(32):=TRUE // underwater menace
  missing(33):=TRUE // moonbase
  missing(34):=TRUE // macra terror
  missing(35):=TRUE // faceless ones
  missing(36):=TRUE // evil of the daleks
  missing(37):=TRUE // tomb of the cybermen
  missing(38):=TRUE // abominable snowmen
  missing(39):=TRUE // ice warriors
  missing(40):=TRUE // enemy of the world
  missing(41):=TRUE // web of fear
  missing(42):=TRUE // fury from the deep
  missing(43):=TRUE // wheel in space
  missing(46):=TRUE // invasion
  missing(49):=TRUE // space pirates
  missing(109):=TRUE // shada
ENDPROC dim'missing
```

more»

Of course, with each missing show easily determined, it was natural to add a printout that listed only the missing shows!

```
PROC print'missing
  PAGE
  PRINT "Now printing list of lost shows..."
  SELECT OUTPUT printer$
  PRINT
  PRINT "Missing Doctor Who Shows:"
  PRINT
  PRINT
  FOR x:=1 TO 49 DO
    IF missing(x) THEN
      read'it(x)
      PRINT USING "###": x;
      PRINT TAB(6),capitalize$(trim$(field$(1
      ))), //wrap line
      PRINT ",";year(x),",";
      PRINT capitalize$(trim$(field$(2)))
      PRINT // blank line
    ENDIF
  ENDFOR x
  PAGE
  SELECT OUTPUT screen$
ENDPROC print'missing
```

Another of my functions is used in print'missing: trim$. It strips off any extra spaces at the end of a string. This is needed when you want to add a comma after a string, for instance. It would look funny if there were a space or two first, then the comma. You also should notice that this function is recursive:

```
FUNC trim$(text$)
  IF text$>"" AND THEN text$(LEN(text$):LEN(
  text$))=" " THEN //wrap line
    RETURN trim$(text$(1:LEN(text$)-1))
  ELSE
    RETURN text$
  ENDIF
ENDFUNC trim$
```

Next, I thought about those who only could print 80 columns. So, I wrote another printout similar to smallprint. I called it generic'print. I took the multi-line per show concept one step further to fit it onto an 80 column chart. I deleted the location column, and print that information directly under the show name. It still looks nice, and is easy to understand.

```
PROC generic'print
  count:=0; pagenumber:=0
  prev'doctor$(1:27):="william hartnell"
  whoyear:=1963
  PAGE
  PRINT "Now printing the chart..."
  SELECT OUTPUT printer$
  read'it(1)
  generic'header
  FOR x:=1 TO last'show DO
    pagetop:=FALSE
    read'it(x)
    whoyear:=year(x)
    generic'check'page
    IF pagetop=TRUE THEN
      generic'page'bottom
      PAGE
      count:=0
      generic'header
      generic'dividing'line
    ELIF field$(2)<>prev'doctor$ OR whoyear<>
    year(x-1) THEN  //wrap line
      generic'doc'header
      generic'bottom'line
    ELSE
      generic'dividing'line
    ENDIF
    generic'small'line
    prev'doctor$:=field$(2)
  ENDFOR x
  generic'page'bottom
  PAGE
  SELECT OUTPUT screen$
ENDPROC generic'print
```

As you may have guessed, this generic printout is heavily based on the smallprint routine. I also quickly modified the header, lines, etc routines to work with the generic 80 column

more»

version. I just tacked on the word generic to the proc names. The differences are minor: shorter dividing lines, different method of checking for a page break, different column titles.

```
PROC generic'small'line
  IF field$(1)(25:27)="   " THEN field$(1)(
  25:27):="("+mark$+")" //wrap line
  PRINT USING "! ###": x,
  IF missing(x) THEN
    PRINT "*",
  ELSE
    PRINT " ",
  ENDIF
  PRINT "!";capitalize$(field$(1));"!";
  PRINT capitalize$(field$(8));"!"
  count:+1
  PRINT "!     !";capitalize$(field$(3));
  PRINT "!";capitalize$(field$(6));"!"
  count:+1
  IF field$(4)<>SPC$(27) OR field$(7)<>SPC$(
  27) THEN //wrap line
    PRINT "!     !";capitalize$(field$(4));
    PRINT "!";capitalize$(field$(7));"!"
    count:+1
  ENDIF
  IF field$(5)<>SPC$(27) THEN
    PRINT "!     !";capitalize$(field$(5));"!";
    PRINT SPC$(27);"!"
    count:+1
  ENDIF
ENDPROC generic'small'line
//
PROC generic'header
  pagenumber:+1
  PRINT 67*"-" //ibm only 67 dashes
  count:+1
  PRINT "! Doctor Who Digest -- Not For",
  PRINT " Publication --";TAB(57);"Page";
  PRINT USING "### !": pagenumber
  count:+1
  generic'doc'header
  generic'title'header
ENDPROC generic'header
//
PROC generic'doc'header
  generic'bottom'line
  PRINT "!";capitalize$(field$(2));
  PRINT TAB(34);whoyear,
  trim'doctor$:=trim$(field$(2))
  PRINT TAB(67-1-LEN(trim'doctor$)),
  PRINT capitalize$(trim'doctor$);"!"
  count:+1
ENDPROC generic'doc'header
//
PROC generic'check'page
  lines'needed:=3+2 // line & footer
  IF field$(4)<>SPC$(27) OR field$(7)<>SPC$(
  27) THEN lines'needed:+1 //wrap line
  IF field$(5)<>SPC$(27) THEN lines'needed:+1
  IF field$(2)<>prev'doctor$ OR year(x)<>year(
  x-1) THEN lines'needed:+2//wrap line
  IF count+lines'needed>=60 THEN pagetop:=TRUE
ENDPROC generic'check'page
//
PROC generic'page'bottom
  generic'bottom'line
  PRINT "!  Copyright 1988 Len Lindsay, 5501";
  PRINT "Groveland, Madison, WI 53716  !"
  PRINT 67*"-" //ibm only 67 dashes
ENDPROC generic'page'bottom
//
PROC generic'dividing'line
  PRINT "!-----!",29*"-","!",29*"-","!"
  //^^^ ibm only, 29 dashes ^^^
  count:+1
ENDPROC generic'dividing'line
//
PROC generic'bottom'line
  PRINT "!"+65*"-"+"!" //ibm only, 65 dashes
  count:+1
ENDPROC generic'bottom'line
//
PROC generic'title'header
  generic'bottom'line
  PRINT "! Show! Name (episodes)//Companions";
  PRINT "! Location & Time//Adversaries!"
  count:+1
ENDPROC generic'title'header
```

There is a special Doctor Who section on QLink (in the Just For Fun section inside

more»

Entertainment). The message board there is one way to find out about what people want. Two things needed were a quick reference list to the shows and a simple checklist of shows (to keep track of video tapes).

First I created a fast reference list. It was easy to do, just a FOR loop for each show, printing the show number, show name, doctor, and year first aired.

```
PROC showlist
  PAGE
  INPUT "ready your printer, hit <return> to
  start: ": reply$ //wrap line
  SELECT OUTPUT printer$
  PRINT "Doctor Who Shows - Fast Guide";
  PRINT "- by Len Lindsay"
  PRINT
  FOR x:=1 TO last'show DO
    PRINT USING "###:": x;
    read'it(x)
    PRINT capitalize$(trim$(field$(1))),",";
    PRINT capitalize$(trim$(field$(2))),",";
    PRINT year(x)
    IF x MOD 57=0 THEN
      PAGE
      PRINT "Doctor Who Shows - Fast Guide";
      PRINT "- continued"
      PRINT
    ENDIF
  ENDFOR x
  PAGE
  SELECT OUTPUT screen$
ENDPROC showlist
```

Next, I took the concept of that list a step further and created a fast checklist. Then decided to make it a little nicer. It now marks the missing shows, and can print the ID for each show if you wish. Once again, page breaks were easy to fit in. Just MOD 55 with the show number, and do a page break whenever the result is 0.

```
PROC check'list
```

```
PAGE
INPUT "Print current ID's in the [  ]? ":
reply$(1:1) //wrap line
blank'id:=FALSE
IF reply$ IN "Nn" THEN blank'id:=TRUE
INPUT "Hit <return> when printer is ready:
 ": reply$ //wrap line
SELECT OUTPUT printer$
PRINT "Doctor Who Shows ID Checklist"
PRINT
FOR x:=1 TO last'show DO
  IF (x MOD 55)=0 THEN check'list'header
  read'it(x)
  IF missing(x) THEN
    PRINT "*",
  ELSE
    PRINT " ",
  ENDIF
  id$:=pack$(id$)
  IF blank'id THEN id$:=""
  IF id$="" THEN
    PRINT "[     ]",
  ELSE
    PRINT "[",
    IF LEN(id$)>3 THEN
      id$(1:5):=id$
      PRINT id$(1:5),
      PRINT "]",
    ELSE
      id$(1:3):=id$
      PRINT " ",id$(1:3);
      PRINT "]",
    ENDIF
  ENDIF
  IF missing(x) THEN
    PRINT "*",
  ELSE
    PRINT " ",
  ENDIF
  PRINT USING "###:": x;
  PRINT capitalize$(trim$(field$(1)));
  PRINT "(",mark$,"),";
  PRINT capitalize$(trim$(field$(2))),",";
  PRINT "(",capitalize$(trim$(field$(8
  ))),")" //wrap line
ENDFOR x
```

more»

```
  PAGE
  SELECT OUTPUT screen$
ENDPROC check'list
//
PROC check'list'header
  PRINT
  PRINT "* means it is a missing show"
  PAGE
  PRINT "Doctor Who Shows ID Checklist";
  PRINT "- continued"
  PRINT
ENDPROC check'list'header
```

Notice that the first two lines of check'list'header actually print a footer on the previous page.

More than half of the check'list routine is dealing with printing the check box at the start of the line. I print a check box with [] square brackets, and 5 spaces inside: [   ]. If there is an ID, I want to print it inside that box. If it more than 5 characters, I ignore all after the first 5. If it is short (like 3 or less characters) I wanted to center it inside the []. These calculations required another function I called pack. It takes all the spaces out of a string. Thus, if the ID was: 18 & 19, it normally wouldn't fit in a 5 space box. Once packed, it fits fine: 18&19.

```
FUNC pack$(text$)
  IF text$="" THEN RETURN ""
  newtext$:=""
  FOR x:=1 TO LEN(text$) DO
    IF text$(x:x)<>" " THEN newtext$:+text$(
    x:x) // wrap line
  ENDFOR x
  RETURN newtext$
ENDFUNC pack$
```

Finally, I wanted to make good use of my LaserJet. I knew it was capable of printing on label stock (full sheets, with peel off backing, to make custom sized labels). So, my challenge was to write the routines to print out labels for the side of a VHS video tape. I wanted the show number and name in **Helvetica**, then the doctors name in **bold**, followed by the location in normal times roman. The next line I wanted the broadcast year in tiny roman. The final line would be in *italic* listing all the companions, followed by the adversaries. It took quite a long time to get this to work just right, since I also had to space it so that it could be cut up by a paper cutter into perfect sized labels (not throwing out anything from between the labels).

I did have about a quarter of the sheet empty on the right side though, as the labels did not go that far over. So, I turned that area into a section for small rectangle labels that could be combined on tapes that held more than one show. Now as I tape the shows from TV, I can label them so that they look like part of the collection they are!

```
PROC vhs'labels
  vhs'prompts
  PRINT "OK ... printing..."
  SELECT OUTPUT printer$
  reset'laserjet
  IF labelstock THEN manual'feed'laserjet
  lpi8
  label'count:=0 //init
  FOR x:=startwith TO endwith DO
    IF NOT missing(x) THEN
      label'count:+1
      IF label'count>13 THEN vhs'next
      label(x)
      FOR z:=1 TO 3 DO PRINT
    ENDIF
  ENDFOR x
  vhs'next; normal'feed'laserjet; roman
  SELECT OUTPUT screen$
ENDPROC vhs'labels
```

This routine is very versatile. It calls another procedure to set up the starting and ending show numbers. Then it checks if it will be printing on regular paper, or label stock (if

more»

label stock, manual feed is used). Then it prints 13 labels per page, using the label routine.

The prompts section has an interesting feature. If you are taping the shows, one per week, it would be entirely reasonable, to just print one sheet of 13 labels every few months. So part of the routine asks for a starting show number, letting you start with any show you wish. Next, if you want a full sheet of labels, just reply 0 as the ending show number; it automatically prints the next 13 shows for you. This is very useful if you are skipping labels for the missing shows (why waste the label paper, you can't tape a show that is not available for broadcast). With the full page option, you do not need to know if any missing shows are included or not, it does it all for you.

```
PROC vhs'prompts
  PAGE
  PRINT "For LaserJet only!!"
  PRINT
  reply$="" // init
  INPUT "skip printing labels for the missing
  shows? ": reply$(1:1) //wrap line
  PRINT
  IF reply$ IN "Nn" THEN
    FOR x:=1 TO last'show DO missing(x
    ):=FALSE //wrap line
  ENDIF
  REPEAT
    PRINT "Start with which show number?";
    PRINT "(1-",last'show,")"
    PRINT "   1",
    INPUT AT 0,1: startwith
    PRINT
  UNTIL startwith>0 AND startwith<=last'show
  REPEAT
    PRINT "End with which show number?";
    PRINT "(",startwith,"-",last'show,")"
    PRINT "[enter 0 for 1 full page of
     labels]" //wrap line
    PRINT "  ";last'show,
    INPUT AT 0,1: endwith
    IF endwith=0 THEN
      temp'count:=0; endwith:=startwith-1
      REPEAT
        endwith:+1
        IF NOT missing(endwith) THEN temp'co
        unt:+1 //wrap line
      UNTIL temp'count>=13 OR endwith>=
      last'show //wrap line
    ENDIF
    PRINT
  UNTIL endwith>=startwith AND endwith<=
  last'show //wrap line
  INPUT "Print on special label stock? ":
  reply$(1:1) //wrap line
  labelstock:=FALSE
  IF reply$ IN "Yy" THEN labelstock:=TRUE
  PRINT
  INPUT "Hit <RETURN> when printer is
  ready:": reply$ //wrap line
ENDPROC vhs'prompts
//
PROC vhs'next
  label'count:=1//the label we will print now
  tiny'roman
  PRINT "Copyright 1988 Len Lindsay, 5501";
  PRINT "Groveland, Madison, WI 53716 ---";
  PRINT "not for duplication or publication"
  PAGE
ENDPROC vhs'next
```

When printing a label, notice that I print both the full size label, and the small rectangle label at the same time. To do this, I use a variable named col2 specifying the amount I need to TAB to get to the start of the second small label.

```
PROC label(shownum)
  col2:=187
  read'it(shownum)
  helvetica
  PRINT shownum,":";capitalize$(trim$(field$(1
  ))); //wrap line
  roman'bold
  PRINT "-",capitalize$(trim$(field$(2)));
  roman
  PRINT "(",capitalize$(trim$(field$(8))),")"
```

more»

```
  // end of first line
  tiny'roman
  PRINT "[",year(shownum),"]",TAB(col2),
  helvetica
  PRINT "Doctor Who - #",shownum
  italic
  PRINT SPC$(6),capitalize$(trim$(field$(3))),
  IF field$(4)<>SPC$(27) THEN PRINT ",";
  capitalize$(trim$(field$(4))), //wrap line
  IF field$(5)<>SPC$(27) THEN PRINT ",";
  capitalize$(trim$(field$(5))), //wrap line
  IF field$(6)<>SPC$(27) THEN PRINT " ::";
  capitalize$(trim$(field$(6))), //wrap line
  IF field$(7)<>SPC$(27) THEN PRINT ",";
  capitalize$(trim$(field$(7))), //wrap line
  PRINT // carriage return
  tiny'roman
  PRINT TAB(col2+3),
  roman'bold
  PRINT capitalize$(trim$(field$(1))),
  roman
ENDPROC label
```

With all these print options, a nice easy way to choose them was needed... a printout MENU. This is a simple menu, no fancy gimmicks. Just list the choices, and INPUT the reply:

```
PROC ask'printout
  PAGE
  PRINT "Printout / Report options:"
  PRINT
  PRINT "F - Fast reference guide to shows"
  PRINT
  PRINT "I - ID checklist (96 col preset)"
  PRINT
  PRINT "E - Epson 100/widepaper(250 col
  preset)" //wrap line
  PRINT
  PRINT "C - Chart of all Shows"
  PRINT "    (LaserJet F, Epson, 128 col
  preset)" //wrap line
  PRINT
  PRINT "V - VHS Labels (LaserJet B, F, or Z)"
  PRINT
  PRINT "M - Missing shows printout"
  PRINT
  PRINT "G - Generic 80 column printer chart"
  PRINT
  PRINT "N - NO printout (same as <return>)"
  PRINT
  PRINT
  INPUT "Your choice? ": reply$(1:1)
  done'printing:=FALSE
  CASE reply$ OF
  WHEN "F","f"
    showlist
  WHEN "I","i"
    check'list
  WHEN "E","e"
    printout
  WHEN "C","c"
    smallprint
  WHEN "G","g","Y","y"
    generic'print
  WHEN "V","v"
    vhs'labels
  WHEN "M","m"
    print'missing
  WHEN "N","n","Q","q"," "
    done'printing:=TRUE
  OTHERWISE
    NULL
  ENDCASE
ENDPROC ask'printout
```

Finally, here are the sections of the program not yet discussed. They generally are quite similar to those in the original program, though you may wish to compare them to find the differences (improvements).

```
PROC start'up
  PAGE
  PRINT "setting up-please wait..."
  dims
  PRINT
  PRINT "These Data Bases are available:"
  set'filename
  record'length:=254
  reply$:="" // initialize
  IF NOT file'exists(filename$) THEN
```

more»

```
  check'file //wrap line
  read'last
  IF last'show<1 THEN add
  current'show:=1 //first one
  dim'missing
  read'it(current'show)
  printer$:="lpt1:"; screen$:="con:" //<<<ibm
  //printer$="lp:";screen$="ds:"//<<<c64
ENDPROC start'up
//
PROC format'screen
  PAGE
  PRINT"+-----------------------------
  -------+" //wrap line
  PRINT "|",
  PRINT SPC$((37-LEN(system'name$)) DIV 2),
  PRINT system'name$+SPC$((37-
  LEN(system'name$))/2), //wrap line
  PRINT "|"
  PRINT "+---------+---------+-----
  ---------+-+" //wrap line
  PRINT "|          |    of    |id:              |
   |" //wrap line
  PRINT "+---------+---------+-------
  -------+-+" //wrap line
  FOR x:=1 TO 8 DO
    PRINT "|          |",SPC$(27),"|"
  ENDFOR x
  PRINT"+---------+-------------------
  -------+" //wrap line
  PRINT AT 4,2: "show num "
  FOR temp:=1 TO 8 DO
    PRINT AT temp+5,2: prompt$(temp)
  ENDFOR temp
ENDPROC format'screen
//
PROC choices
  CURSOR 14,1
  PRINT"+------+--+------------------
  ------+--+" //wrap line
  FOR x:=1 TO 8 DO
    PRINT "       |  |",SPC$(25),"|"
  ENDFOR x
  PRINT"       +--+-------------------------+"
  CURSOR 15,1
  PRINT AT CURROW,9: " a"
```

```
PRINT AT CURROW,9: " e"
PRINT AT CURROW,9: " b"
PRINT AT CURROW,9: " n"
PRINT AT CURROW,9: " p"
PRINT AT CURROW,9: "##"
PRINT AT CURROW,9: " s"
PRINT AT CURROW,9: " q"
CURSOR 15,1
PRINT AT CURROW,12: "add shows to list"
PRINT AT CURROW,12: "edit this show"
PRINT AT CURROW,12: "browse (autoviewer)"
PRINT AT CURROW,12: "next show (+)"
PRINT AT CURROW,12: "previous show (-)"
PRINT AT CURROW,12: "display this show #"
PRINT AT CURROW,12: "search"
PRINT AT CURROW,12: "quit";
IF filename$="docwho.ran" THEN
  PRINT "(see printout menu)"
ELSE
  PRINT // cr
ENDIF
PRINT
INPUT AT CURROW,1,3: "your choice: ":
reply$, //wrap line
clear'choices
done:=FALSE
CASE reply$ OF
WHEN "a","A"
  add
WHEN "e","E"
  edit'
WHEN "b","B"
  browse
WHEN "n","N","+"
  next'show
WHEN "p","P","-"
  previous'show
WHEN "s","S"
  search
WHEN "q","Q"
  done:=TRUE
OTHERWISE
  temp:=value(reply$)
  IF temp>0 AND temp<=last'show THEN
    current'show:=temp
    read'it(current'show)
```

more»

```
    ENDIF
  ENDCASE
  pause(1)
  clear'keys
ENDPROC choices
//
PROC display'
  PRINT AT 4,2: "show num "
  PRINT AT 4,12: USING "###": current'show
  PRINT AT 4,19: USING "###": last'show
  PRINT AT 4,26: id$
  PRINT AT 4,38: mark$
  CURSOR 6,1
  FOR temp:=1 TO 8 DO
    PRINT AT CURROW,12: field$(temp)
  ENDFOR temp
ENDPROC display'
//
PROC input'data
  CURSOR 6,1
  FOR temp:=1 TO 8 DO
    INPUT AT CURROW,12,27: "":
    temp'input$(1:27)//wrap line
    IF LEN(temp'input$)>0 THEN field$(
    temp):=temp'input$ //wrap line
  ENDFOR temp
  INPUT AT CURROW,12,11: "": temp'input$
  IF LEN(temp'input$)>0 THEN id$:=temp'input$
  INPUT AT CURROW,12,1: "": temp'input$
  IF LEN(temp'input$)>0 THEN mark$:=
  temp'input$ //wrap line
ENDPROC input'data
//
PROC add
  current'show:=last'show
  done'adding:=FALSE
  REPEAT
    mark$:=" "
    id$:=SPC$(11)
    FOR temp:=1 TO 8 DO field$(temp):=SPC$(27)
    format'screen
    PRINT AT 4,2: "adding no",
    PRINT AT CURROW,12: USING "###       ":
    current'show+1 //wrap line
    display'bottom(FALSE) //empty
    REPEAT
      input'data
      add'status
    UNTIL data'ok OR done'adding
    IF data'ok THEN
      current'show:+1
      last'show:+1 //record accepted
      IF write'record(last'show) THEN
        write'last
      ELSE
        done'adding:=TRUE
        last'show:-1 //none added
        current'show:-1 //none added
        read'it(current'show) //refresh values
      ENDIF
    ELSE // abort
      read'it(current'show) // refresh values
    ENDIF
  UNTIL done'adding
ENDPROC add
//
PROC edit'
  REPEAT
    display'
    display'bottom(TRUE)
    input'data
    edit'status
  UNTIL data'ok OR done'editing
  IF data'ok THEN
    IF NOT write'record(current'show)
    THEN read'it(current'show) //wrap line
  ELSE // aborted
    read'it(current'show) //refresh values
  ENDIF
ENDPROC edit'
//
PROC next'show
  IF current'show<last'show
  THEN current'show:+1 //wrap line
  read'it(current'show)
ENDPROC next'show
//
PROC previous'show
  IF current'show>1 THEN current'show:-1
  read'it(current'show)
ENDPROC previous'show
//
```

more»

```
PROC dims
  DIM system'name$ OF 37
  DIM prompt$(1:10) OF 9
  DIM field$(1:8) OF 27
  DIM find'field$(1:8) OF 27
  DIM match$(1:8) OF 8
  DIM id$ OF 11, find'id$ OF 11
  DIM mark$ OF 1, find'mark$ OF 1
  DIM text$ OF 27, newtext$ OF 27
  DIM temp'input$ OF 27, doctor$ OF 27
  DIM trim'doctor$ OF 27, prev'doctor$ OF 27
  DIM episode$ OF 3, prev'char$ OF 1
  DIM printer$ OF 5, screen$ OF 4
  DIM reply$ OF 27, lstart$ OF 2
  DIM continue$ OF 1
ENDPROC dims
//
FUNC file'exists(filename$) CLOSED
  TRAP
    OPEN FILE 7,filename$,READ
    CLOSE FILE 7
    RETURN TRUE
  HANDLER
    RETURN FALSE
  ENDTRAP
ENDFUNC file'exists
//
PROC clear'choices
  CURSOR 14,1
  PRINT "+---------+-------------------
  --------+" //wrap line
  FOR lines:=15 TO 24 DO
    clear'line(lines)
  ENDFOR lines
ENDPROC clear'choices
//
PROC halt
  CLOSE
  PAGE
  END "finished"
ENDPROC halt
//
PROC find'input
  format'screen
  display'bottom(FALSE)
  PRINT AT 4,12: "all"
  PRINT AT 4,19: USING "###": last'show
  PRINT AT 4,27: "searching:"
  PRINT AT 23,1: " enter text to search for"
  PRINT AT 24,1: " UPPER or lower case
  doesn't matter" //wrap line
  CURSOR 6,1
  FOR temp:=1 TO 8 DO
    INPUT AT CURROW,12,27: "": find'f
    ield$(temp) //wrap line
    // all lower case search
    find'field$(temp):=lower$(find'field$(
    temp)) //wrap line
  ENDFOR temp
  INPUT AT CURROW,12,11: "": find'id$
  INPUT AT CURROW,12,1: "": find'mark$
  clear'line(23); clear'line(24)
ENDPROC find'input
//
PROC search
  find'input
  TRAP ESC-
  open'it
  searching:=0
  continue$:="" //keep searching
  PRINT AT 24,1: " press <ctrl>+<break> to
   quit search        ", //wrap line
  REPEAT
    searching:+1
    PRINT AT 4,12: USING "###": searching
    read'record(searching)
    match'record
    IF matching THEN
      clear'choices
      close'it
      current'show:=searching
      display'
      INPUT AT 24,1,1: " <return> to continue
      or a to abort: ": continue$, //wrap line
      IF continue$="" THEN
        PRINT AT 24,1: " press <ctrl>+<break>
        to quit search        ", //wrap line
        open'it
      ENDIF
    ENDIF
  UNTIL ESC OR (searching=last'show) OR
  (continue$>"") //wrap line
```

more»

```
  close'it
  TRAP ESC+
  IF searching<>current'show THEN read'it(
  current'show) //last found//wrap line
ENDPROC search
//
PROC clear'line(line)
  PRINT AT line,1: SPC$(40),
ENDPROC clear'line
//
PROC browse
  PRINT AT 24,1: "how many seconds delay
  between shows:  ", //wrap line
  INPUT AT 24,38,2: "": delay,
  format'screen
  PRINT AT 24,1: " hit <ctrl>+<break> to quit
  browsing        ", //wrap line
  TRAP ESC- //disable stop key
  open'it
  REPEAT
    IF current'show<last'show
    THEN current'show:+1 //wrap line
    read'record(current'show)
    display'
    pause(delay)
  UNTIL current'show=last'show OR ESC
  TRAP ESC+
  close'it
ENDPROC browse
//
PROC pause(seconds)
  cycles:=2000 //<<<change for your computer
  FOR now:=1 TO seconds*cycles DO NULL
ENDPROC pause
//
PROC add'status
  status'outline
  REPEAT
    INPUT AT 23,2,1: "what is your command: ":
    reply$(1:1), //wrap line
  UNTIL reply$ IN " nyda"
  clear'status
  CASE reply$ OF
  WHEN " ","y","Y"
    data'ok:=TRUE
  WHEN "n","N"
    data'ok:=FALSE
  WHEN "d","D"
    done'adding:=TRUE
    data'ok:=TRUE
  WHEN "a","A"
    done'adding:=TRUE
    data'ok:=FALSE
  OTHERWISE
    data'ok:=FALSE // shouldnt be here
  ENDCASE
ENDPROC add'status
//
PROC edit'status
  REPEAT
    INPUT AT 24,1,1: "data ok? y=yes n=no
    a=abort: ": reply$(1:1), //wrap line
  UNTIL reply$ IN " nya"
  clear'line(24)
  done'editing:=FALSE
  CASE reply$ OF
  WHEN " ","y","Y"
    data'ok:=TRUE
  WHEN "n","N"
    data'ok:=FALSE
  WHEN "a","A"
    done'editing:=TRUE
    data'ok:=FALSE
  OTHERWISE
    data'ok:=FALSE // shouldnt be here
  ENDCASE
ENDPROC edit'status
//
PROC display'bottom(show'data)
  PRINT AT 14,1:"|",prompt$(9),"|          +"
  PRINT AT CURROW,1: "|",prompt$(10)
  ,"| +---------+" //wrap line
  PRINT "+---------+-+"
  IF show'data THEN
    PRINT AT 14,12: id$
    PRINT AT 15,12: mark$
  ENDIF
ENDPROC display'bottom
//
PROC status'outline
  CURSOR 16,1
  PRINT "+---------+-+----------------
```

more»

```
  ---------+" //wrap line
  FOR x:=1 TO 5 DO
    PRINT "|         |",SPC$(27),"|"
  ENDFOR x
  PRINT "+---------+------------------
  -------+" //wrap line
  PRINT "|",SPC$(37),"|"
  PRINT "+------------------------------
  ------+", //wrap line
  PRINT AT 17,2: "<return>"
  PRINT AT 18,9: "y"
  PRINT AT 19,9: "n"
  PRINT AT 20,9: "d"
  PRINT AT 21,9: "a"
  PRINT AT 17,13: "default - same as y"
  PRINT AT 18,13: "yes, data ok - do next one"
  PRINT AT 19,13: "no, redo data input"
  PRINT AT 20,13: "done - save and end input"
  PRINT AT 21,13: "abort - stop - don't save"
ENDPROC status'outline
//
PROC clear'status
  CURSOR 16,1
  PRINT "+---------+-+",SPC$(26),
  FOR lines:=17 TO 24 DO clear'line(lines)
ENDPROC clear'status
//
PROC clear'keys
  WHILE KEY$>"" DO NULL
ENDPROC clear'keys
//
PROC check'file
  REPEAT
    PAGE
    PRINT "data file";filename$;"not found..."
    PRINT AT 7,1: " ======================"
    PRINT AT 9,1: "insert disk with file,
    then:" //wrap line
    set'filename
    TRAP
      MOUNT
    HANDLER
      NULL
    ENDTRAP
  UNTIL file'exists(filename$)
ENDPROC check'file
```

Note, that the information stored in the records is one of the key items! That information takes some time to type in, and much longer to collect. If you want the information, ready to use on disk, the original Doctor Who data is on Today Disk 15, the Star Trek data is on Today Disk 16, and I hope to be able to put the new expanded Doctor Who data on Today Disk 23. ■

¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤

Create the Database File

Creating the random file for the database is easy. Filling it with data is what takes time. The **CREATE** statement creates the file, and sets up as many blank records as you specify. When typing in data in a new database, creating the records as you go takes longer than creating a whole bunch of blank records and just filling them in. So, to save yourself time, calculate about how many records you will need, and use that number in the **CREATE** statement. For Doctor Who, initially set up the file for 155 shows. Later you can expand it for more shows as they are broadcast.

```
INPUT "Filename:": filename$
INPUT "How many shows to start with:": shows
record'length:=254; last'show:=0
CREATE filename$,shows,record'length
OPEN FILE 2,filename$,RANDOM record'length
WRITE FILE 2,1: last'show
CLOSE
```

After creating the database file, you need to run the Edit'prompts program (see the next page). This program will set up the prompts needed for the database as well as the compatible fields for the search option. ■

Edit Prompts

by Len Lindsay

This short program shows how to edit prompts (stored in the first record of the database). I stole as much as I could from the database program. The main program just calls the few procedures one after another. Take a look and you will see the sections I borrowed:

- **start'up** is condensed
- **dims** is condensed
- **read'last** is expanded
  The original database random file only stored the number of the last show in the first record. The rest of the record was blank. In case there are no prompts there to edit, I added a TRAP for catching the error trying to read them. The HANDLER then sets prompt$ array to 9 spaces and match$ arrays to nulls.
- **format'screen** is the same
- **display'** is shortened (only the part needed)
- **display'bottom** is shortened (only part needed)
- **write'last** is the same
- **open'it** is the same
- **close'it** is the same
- **input'prompts** is added
- **ask'matching** is added
- **instruct** is added

Only three procedures are new! A few are shortened (I deleted lines not needed). One was extended (read'last to allow use of old database files). One new procedure is only two lines, to erase screen line 18, then print a message there. Another procedure simply inputs the prompts you want (the name of the system too). Note, for Doctor Who, the compatible fields are:

1 - show name (none, just hit «return»)
2 - doctor name (none, just hit «return»)
3 - companion (enter: 345)
4 - companion (enter: 345)
5 - companion (enter: 345)
6 - adversary (enter: 67)
7 - adversary (enter: 67)
8 - location (none, just hit «return»)

```
start'up
format'screen
display'
display'bottom
input'prompts
ask'matching
write'last
//
PROC open'it
 OPEN FILE 2,filename$,RANDOM record'length
ENDPROC open'it
//
PROC close'it
  TRAP
    CLOSE FILE 2
  HANDLER
  ENDTRAP
ENDPROC close'it
//
PROC read'last
  open'it
  READ FILE 2,1: last'show
  TRAP
    READ FILE 2: system'name$
    FOR temp:=1 TO 10 DO
      READ FILE 2: prompt$(temp)(1:9)
    ENDFOR temp
    FOR temp:=1 TO 8 DO
      READ FILE 2: match$(temp)
    ENDFOR temp
  HANDLER//only number in file
    FOR x:=1 TO 10 DO prompt$(x):=SPC$(9)
    FOR x:=1 TO 8 DO match$(x):=""
  ENDTRAP
  close'it
ENDPROC read'last
//
PROC write'last
  open'it
  WRITE FILE 2,1: last'show
  WRITE FILE 2: system'name$
  FOR temp:=1 TO 10 DO
    WRITE FILE 2: prompt$(temp)
  ENDFOR temp
  FOR temp:=1 TO 8 DO
    WRITE FILE 2: match$(temp)
```

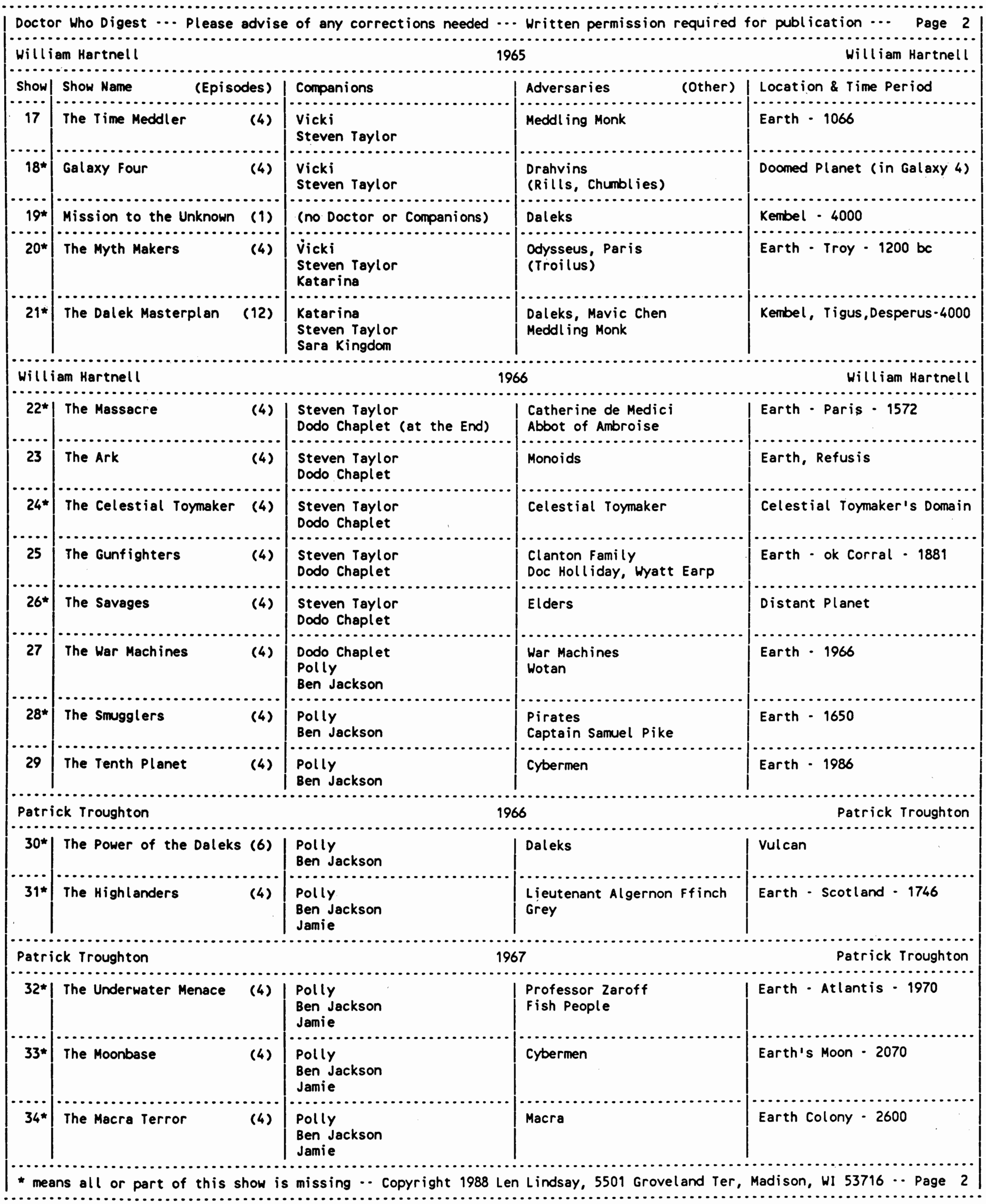
more»

Edit Prompts - continued

```
  ENDFOR temp
  close'it
ENDPROC write'last
//
PROC dims
  DIM system'name$ OF 37
  DIM prompt$(1:10) OF 9
  DIM match$(1:8) OF 8
  DIM id$ OF 11
  DIM mark$ OF 1
ENDPROC dims
//
PROC format'screen
  PAGE
  PRINT "+-----------------------------
  -------+" //wrap line
  PRINT "|",
  PRINT SPC$((37-LEN(system'name$)) DIV 2),
  PRINT system'name$+SPC$((37-LEN(system'
  name$))/2), //wrap line
  PRINT "|"
  PRINT "+---------+----------+----------
  ----+-+" //wrap line
  PRINT "|         |    of    |id:              |
   |" //wrap line
  PRINT "+---------+----------+-----------
  ---+-+" //wrap line
  FOR x:=1 TO 8 DO
    PRINT "|         |",SPC$(27),|"
  ENDFOR x
  PRINT "+---------+---------------------
  -----+" //wrap line
  PRINT AT 4,2: "show num "
  FOR temp:=1 TO 8 DO
    PRINT AT temp+5,2: prompt$(temp)
  ENDFOR temp
ENDPROC format'screen
//
PROC display'
  PRINT AT 4,2: "show num "
  PRINT AT 4,12: USING "###": current'show
  PRINT AT 4,19: USING "###": last'show
ENDPROC display'
//
PROC display'bottom
  PRINT AT 14,1: "|",prompt$(9),"|              +"
  PRINT AT CURROW,1: "|",prompt$(10),"| +----
  -----+" //wrap line
  PRINT "+---------+-+"
ENDPROC display'bottom
//
PROC start'up
  PAGE
  PRINT "setting up-please wait..."
  dims
  PRINT "Filename: docwho.rnd",
  INPUT AT 0,11:"":filename$
  record'length:=254
  read'last
  current'show:=1
ENDPROC start'up
//
PROC input'prompts
  instruct("Enter the name of the database
   system (37 characters)") //wrap line
  INPUT AT 2,2,37: "": system'name$
  WHILE LEN(system'name$)>1 AND THE
  N system'name$(1:1)=" " DO //wrap line
    system'name$:=system'name$(2:LEN(syst
    em'name$)) //wrap line
  ENDWHILE
  FOR x:=1 TO 10 DO
    instruct("Enter prompt number "+STR$(x
    )+" (up to 9 characters)")//wrap line
    INPUT AT 5+x,2,9: "": prompt$(x)
  ENDFOR x
  instruct("")
ENDPROC input'prompts
//
PROC instruct(text$)
  PRINT AT 18,1: SPC$(79)
  PRINT AT 18,1: text$
ENDPROC instruct
//
PROC ask'matching
  FOR temp:=1 TO 8 DO
    PRINT "What other fields are compatible"
    PRINT "with field";temp;prompt$(temp)
    INPUT match$(temp)
  ENDFOR temp
ENDPROC ask'matching ■
```

Sample Chart

The chart printed by the Database program was nearly an inch too wide for a newsletter page. The sample shown here was reduced to fit.



```
.....................................................................................................................
| Doctor Who Digest --- Please advise of any corrections needed --- Written permission required for publication ---   Page  2 |
|....................................................................................................................|
| William Hartnell                                      1965                                        William Hartnell |
|....................................................................................................................|
| Show| Show Name          (Episodes) | Companions                 | Adversaries             (Other) | Location & Time Period     |
|.....|...............................|............................|.................................|............................|
|  17 | The Time Meddler          (4) | Vicki                      | Meddling Monk                   | Earth - 1066               |
|     |                               | Steven Taylor              |                                 |                            |
|.....|...............................|............................|.................................|............................|
|  18*| Galaxy Four               (4) | Vicki                      | Drahvins                        | Doomed Planet (in Galaxy 4) |
|     |                               | Steven Taylor              | (Rills, Chumblies)              |                            |
|.....|...............................|............................|.................................|............................|
|  19*| Mission to the Unknown    (1) | (no Doctor or Companions)  | Daleks                          | Kembel - 4000              |
|.....|...............................|............................|.................................|............................|
|  20*| The Myth Makers           (4) | Vicki                      | Odysseus, Paris                 | Earth - Troy - 1200 bc     |
|     |                               | Steven Taylor              | (Troilus)                       |                            |
|     |                               | Katarina                   |                                 |                            |
|.....|...............................|............................|.................................|............................|
|  21*| The Dalek Masterplan     (12) | Katarina                   | Daleks, Mavic Chen              | Kembel, Tigus,Desperus-4000 |
|     |                               | Steven Taylor              | Meddling Monk                   |                            |
|     |                               | Sara Kingdom               |                                 |                            |
|....................................................................................................................|
| William Hartnell                                      1966                                        William Hartnell |
|....................................................................................................................|
|  22*| The Massacre              (4) | Steven Taylor              | Catherine de Medici             | Earth - Paris - 1572       |
|     |                               | Dodo Chaplet (at the End)  | Abbot of Ambroise               |                            |
|.....|...............................|............................|.................................|............................|
|  23 | The Ark                   (4) | Steven Taylor              | Monoids                         | Earth, Refusis             |
|     |                               | Dodo Chaplet               |                                 |                            |
|.....|...............................|............................|.................................|............................|
|  24*| The Celestial Toymaker    (4) | Steven Taylor              | Celestial Toymaker              | Celestial Toymaker's Domain |
|     |                               | Dodo Chaplet               |                                 |                            |
|.....|...............................|............................|.................................|............................|
|  25 | The Gunfighters           (4) | Steven Taylor              | Clanton Family                  | Earth - ok Corral - 1881   |
|     |                               | Dodo Chaplet               | Doc Holliday, Wyatt Earp        |                            |
|.....|...............................|............................|.................................|............................|
|  26*| The Savages               (4) | Steven Taylor              | Elders                          | Distant Planet             |
|     |                               | Dodo Chaplet               |                                 |                            |
|.....|...............................|............................|.................................|............................|
|  27 | The War Machines          (4) | Dodo Chaplet               | War Machines                    | Earth - 1966               |
|     |                               | Polly                      | Wotan                           |                            |
|     |                               | Ben Jackson                |                                 |                            |
|.....|...............................|............................|.................................|............................|
|  28*| The Smugglers             (4) | Polly                      | Pirates                         | Earth - 1650               |
|     |                               | Ben Jackson                | Captain Samuel Pike             |                            |
|.....|...............................|............................|.................................|............................|
|  29 | The Tenth Planet          (4) | Polly                      | Cybermen                        | Earth - 1986               |
|     |                               | Ben Jackson                |                                 |                            |
|....................................................................................................................|
| Patrick Troughton                                     1966                                       Patrick Troughton |
|....................................................................................................................|
|  30*| The Power of the Daleks   (6) | Polly                      | Daleks                          | Vulcan                     |
|     |                               | Ben Jackson                |                                 |                            |
|.....|...............................|............................|.................................|............................|
|  31*| The Highlanders           (4) | Polly                      | Lieutenant Algernon Ffinch      | Earth - Scotland - 1746    |
|     |                               | Ben Jackson                | Grey                            |                            |
|     |                               | Jamie                      |                                 |                            |
|....................................................................................................................|
| Patrick Troughton                                     1967                                       Patrick Troughton |
|....................................................................................................................|
|  32*| The Underwater Menace     (4) | Polly                      | Professor Zaroff                | Earth - Atlantis - 1970    |
|     |                               | Ben Jackson                | Fish People                     |                            |
|     |                               | Jamie                      |                                 |                            |
|.....|...............................|............................|.................................|............................|
|  33*| The Moonbase              (4) | Polly                      | Cybermen                        | Earth's Moon - 2070        |
|     |                               | Ben Jackson                |                                 |                            |
|     |                               | Jamie                      |                                 |                            |
|.....|...............................|............................|.................................|............................|
|  34*| The Macra Terror          (4) | Polly                      | Macra                           | Earth Colony - 2600        |
|     |                               | Ben Jackson                |                                 |                            |
|     |                               | Jamie                      |                                 |                            |
|....................................................................................................................|
| * means all or part of this show is missing -- Copyright 1988 Len Lindsay, 5501 Groveland Ter, Madison, WI 53716 -- Page  2 |
.....................................................................................................................
```

Doctor Who Shows - Fast Guide - (modified)

1: An Unearthly Child, William Hartnell, 1963
2: The Daleks, William Hartnell, 1963
3: The Edge of Destruction, William Hartnell, 1964
4: Marco Polo, William Hartnell, 1964
5: The Keys of Marinus, William Hartnell, 1964
6: The Aztecs, William Hartnell, 1964
7: The Sensorites, William Hartnell, 1964
8: The Reign of Terror, William Hartnell, 1964
9: Planet of Giants, William Hartnell, 1964
10: The Dalek Invasion of Earth, William Hartnell, 1964
11: The Rescue, William Hartnell, 1965
12: The Romans, William Hartnell, 1965
13: The Web Planet, William Hartnell, 1965
14: The Crusade, William Hartnell, 1965
15: The Space Museum, William Hartnell, 1965
16: The Chase, William Hartnell, 1965
17: The Time Meddler, William Hartnell, 1965
18: Galaxy Four, William Hartnell, 1965
19: Mission to the Unknown, William Hartnell, 1965
20: The Myth Makers, William Hartnell, 1965
21: The Dalek Masterplan, William Hartnell, 1965
22: The Massacre, William Hartnell, 1966
23: The Ark, William Hartnell, 1966
24: The Celestial Toymaker, William Hartnell, 1966
25: The Gunfighters, William Hartnell, 1966
26: The Savages, William Hartnell, 1966
27: The War Machines, William Hartnell, 1966
28: The Smugglers, William Hartnell, 1966
29: The Tenth Planet, William Hartnell, 1966
30: The Power of the Daleks, Patrick Troughton, 1966
31: The Highlanders, Patrick Troughton, 1966
32: The Underwater Menace, Patrick Troughton, 1967
33: The Moonbase, Patrick Troughton, 1967
34: The Macra Terror, Patrick Troughton, 1967
35: The Faceless Ones, Patrick Troughton, 1967
36: The Evil of the Daleks, Patrick Troughton, 1967
37: The Tomb of the Cybermen, Patrick Troughton, 1967
38: The Abominable Snowmen, Patrick Troughton, 1967
39: The Ice Warriors, Patrick Troughton, 1967
40: The Enemy of the World, Patrick Troughton, 1967
41: The Web of Fear, Patrick Troughton, 1968
42: Fury From the Deep, Patrick Troughton, 1968
43: The Wheel in Space, Patrick Troughton, 1968
44: The Dominators, Patrick Troughton, 1968
45: The Mind Robber, Patrick Troughton, 1968
46: The Invasion, Patrick Troughton, 1968
47: The Krotons, Patrick Troughton, 1968
48: The Seeds of Death, Patrick Troughton, 1969
49: The Space Pirates, Patrick Troughton, 1969
50: The War Games, Patrick Troughton, 1969
51: Spearhead From Space, Jon Pertwee, 1970
52: Dr. Who and the Silurians, Jon Pertwee, 1970
53: The Ambassadors of Death, Jon Pertwee, 1970
54: Inferno, Jon Pertwee, 1970
55: Terror of the Autons, Jon Pertwee, 1971
56: The Mind of Evil, Jon Pertwee, 1971
57: The Claws of Axos, Jon Pertwee, 1971
58: Colony in Space, Jon Pertwee, 1971
59: The Daemons, Jon Pertwee, 1971
60: The Day of the Daleks, Jon Pertwee, 1972
61: The Curse of Peladon, Jon Pertwee, 1972
62: The Sea Devils, Jon Pertwee, 1972
63: The Mutants, Jon Pertwee, 1972
64: The Time Monster, Jon Pertwee, 1972
65: The Three Doctors, Pertwee Hartnell Troughton, 1972
66: Carnival of Monsters, Jon Pertwee, 1973
67: Frontier in Space, Jon Pertwee, 1973
68: Planet of the Daleks, Jon Pertwee, 1973
69: The Green Death, Jon Pertwee, 1973
70: The Time Warrior, Jon Pertwee, 1973
71: Invasion of the Dinosaurs, Jon Pertwee, 1974
72: Death to the Daleks, Jon Pertwee, 1974
73: The Monster of Peladon, Jon Pertwee, 1974
74: Planet of the Spiders, Jon Pertwee, 1974
75: Robot, Tom Baker, 1974
76: The Ark in Space, Tom Baker, 1975
77: The Sontaran Experiment, Tom Baker, 1975

Doctor Who Shows - Fast Guide - continued

78: Genesis of the Daleks, Tom Baker, 1975
79: Revenge of the Cybermen, Tom Baker, 1975
80: Terror of the Zygons, Tom Baker, 1975
81: Planet of Evil, Tom Baker, 1975
82: Pyramids of Mars, Tom Baker, 1975
83: The Android Invasion, Tom Baker, 1975
84: The Brain of Morbius, Tom Baker, 1976
85: The Seeds of Doom, Tom Baker, 1976
86: The Masque of Mandragora, Tom Baker, 1976
87: The Hand of Fear, Tom Baker, 1976
88: The Deadly Assassin, Tom Baker, 1976
89: Face of Evil, Tom Baker, 1977
90: The Robots of Death, Tom Baker, 1977
91: The Talons of Weng-Chiang, Tom Baker, 1977
92: Horror of Fang Rock, Tom Baker, 1977
93: The Invisible Enemy, Tom Baker, 1977
94: Image of the Fendahl, Tom Baker, 1977
95: The Sun Makers, Tom Baker, 1977
96: Underworld, Tom Baker, 1978
97: The Invasion of Time, Tom Baker, 1978
98: Ribos Operation (Keys1), Tom Baker, 1978
99: The Pirate Planet (Keys2), Tom Baker, 1978
100: The Stones of Blood (Keys3), Tom Baker, 1978
101: The Androids of Tara(Keys4), Tom Baker, 1978
102: The Power of Kroll (Keys5), Tom Baker, 1978
103: Armageddon Factor (Keys6), Tom Baker, 1979
104: Destiny of the Daleks, Tom Baker, 1979
105: City of Death, Tom Baker, 1979
106: The Creature From the Pit, Tom Baker, 1979
107: Nightmare of Eden, Tom Baker, 1979
108: The Horns of Nimon, Tom Baker, 1979
109: Shada (never completed), Tom Baker, 1980
110: The Leisure Hive, Tom Baker, 1980
111: Meglos, Tom Baker, 1980
112: Full Circle, Tom Baker, 1980
113: State of Decay, Tom Baker, 1980
114: Warriors' Gate, Tom Baker, 1981
115: The Keeper of Traken, Tom Baker, 1981
116: Logopolis, Tom Baker, 1981
117: Castrovalva, Peter Davison, 1982
118: Four to Doomsday, Peter Davison, 1982
119: Kinda, Peter Davison, 1982
120: The Visitation, Peter Davison, 1982
121: Black Orchid, Peter Davison, 1982
122: Earthshock, Peter Davison, 1982
123: Timeflight, Peter Davison, 1982
124: Arc of Infinity, Peter Davison, 1983
125: Snakedance, Peter Davison, 1983
126: Mawdryn Undead, Peter Davison, 1983
127: Terminus, Peter Davison, 1983
128: Enlightenment, Peter Davison, 1983
129: King's Demons, Peter Davison, 1983
130: The Five Doctors, Davison Pertwee Troughton.., 1983
131: Warriors of the Deep, Peter Davison, 1984
132: The Awakening, Peter Davison, 1984
133: Frontios, Peter Davison, 1984
134: Resurection of the Daleks, Peter Davison, 1984
135: Planet of Fire, Peter Davison, 1984
136: The Caves of Androzani, Peter Davison, 1984
137: The Twin Dilemma, Colin Baker, 1984
138: Attack of the Cybermen, Colin Baker, 1984
139: Vengence on Varos, Colin Baker, 1985
140: Mark of the Rani, Colin Baker, 1985
141: The Two Doctors, Colin Baker, Pat Troughton, 1985
142: Timelash, Colin Baker, 1985
143: Revelation of the Daleks, Colin Baker, 1985
144: Mysterious Planet (Trial1), Colin Baker, 1986
145: Mindwarp (Trial2), Colin Baker, 1986
146: Terror of Vervoids (Trial3), Colin Baker, 1986
147: The Ultimate Foe (Trial4), Colin Baker, 1986
148: Time and the Rani, Sylvester McCoy, 1987
149: Paradise Towers, Sylvester McCoy, 1987
150: Delta and the Bannermen, Sylvester McCoy, 1987
151: Dragonfire, Sylvester McCoy, 1987
152: Remembrance of the Daleks, Sylvester McCoy, 1988
153: The Greatest Show in Galaxy, Sylvester McCoy, 1988
154: Happiness Patrol, Sylvester McCoy, 1989

Sample Output

The VHS labels produced by the Database program were about an inch too wide for a newsletter page. The sample shown is reduced.

67: Frontier in Space -**Jon Pertwee** (Earth & Moon - 2500)
[1973]
Jo Grant :: The Master, Draconians, Ogrons, Daleks

Doctor Who - #67
Frontier in Space

68: Planet of the Daleks -**Jon Pertwee** (Spiridon - 2500)
[1973]
Jo Grant :: Daleks, Spiridons

Doctor Who - #68
Planet of the Daleks

69: Green Death -**Jon Pertwee** (Metebelis 3, Earth - 1980's)
[1973]
Jo Grant, Unit :: Green Maggots, Green Slime, Boss

Doctor Who - #69
Green Death

70: Time Warrior -**Jon Pertwee** (Earth: Midieval & 1980's)
[1973]
Sarah Jane Smith, Unit :: Sontaran, Irongron, Robot Knight

Doctor Who - #70
Time Warrior

71: Invasion of the Dinosaurs -**Jon Pertwee** (Earth - England - 1980's)
[1974]
Sarah Jane Smith, Unit :: Dinosaurs, Captain Yates

Doctor Who - #71
Invasion of the Dinosaurs

72: Death to the Daleks -**Jon Pertwee** (Exxilon - 2800)
[1974]
Sarah Jane Smith :: Daleks, the City

Doctor Who - #72
Death to the Daleks

73: Monster of Peladon -**Jon Pertwee** (Peladon - 3550)
[1974]
Sarah Jane Smith :: Ice Warriors (Gebek), Ekersley, Aggedor

Doctor Who - #73
Monster of Peladon

74: Planet of the Spiders -**Jon Pertwee** (Metebelis 3, England 1980's)
[1974]
Sarah Jane Smith, Unit :: Great One, Lupton, Giant Spiders

Doctor Who - #74
Planet of the Spiders

75: Robot -**Tom Baker** (Earth - England - 1980's)
[1974]
Sarah Jane Smith, Harry Sullivan, Unit :: Giant Robot, Hilda Winters

Doctor Who - #75
Robot

76: Ark in Space -**Tom Baker** (Ark Sp. Station Nerva 4300)
[1975]
Sarah Jane Smith, Harry Sullivan :: Wirrn

Doctor Who - #76
Ark in Space

77: Sontaran Experiment -**Tom Baker** (Earth - 4300)
[1975]
Sarah Jane Smith, Harry Sullivan :: Sontarans, Styre's Robot

Doctor Who - #77
Sontaran Experiment

78: Genesis of the Daleks -**Tom Baker** (Skaro)
[1975]
Sarah Jane Smith, Harry Sullivan :: Daleks, Davros, (Kaleds, Thals)

Doctor Who - #78
Genesis of the Daleks

79: Revenge of the Cybermen -**Tom Baker** (Sp.Station Nerva, Voga 2900)
[1975]
Sarah Jane Smith, Harry Sullivan :: Cybermats, Cybermen

Doctor Who - #79
Revenge of the Cybermen

How To Submit

More and more computer systems now support COMAL. This makes it harder to do this newsletter. Articles and programs are needed, especially relating to the newest COMAL implementations. If you send in a program, put it on your disk twice:

SAVE "name"
LIST "name.lst"

Also, if possible include a short (or long if you wish) article about the program. Put the article on the same disk as a standard SEQ text file. Also include a printout of the article if you can (no need to send a printout of the program listing though).

Include your name and subscriber number on the disk label as well as in the first line of the program and article. Also put the computer type on the disk label so I know where to start with it. Eventually, all text and listings end up on my IBM PC hard disk. I use Big Blue Reader to transfer disks from C64 to IBM. Amiga will be a different story. It looks like I will need to invest in an Amiga 2000 with IBM PC card. Will there be enough interest (sales) to cover the equipment cost?

Send it to our new address: COMAL Users Group USA Ltd, 5501 Groveland, Madison, WI 53716. Material submitted is not returned.

Break Away From Reality

INFO magazine has *The Real World* column in each issue to remind their readers that there is more to life than computers. I'd like to remind all COMALites that there is more than just reality too. Take a break from COMAL. Take a break from reality. Watch a Doctor Who show. If you haven't seen it yet, here is a list of stations that broadcast the show (I got this information via QLink). I find the shows very enjoyable. If you watch Doctor Who and your local station is not on my list, please send me a postcard with the info on it. I'd like to keep my list up to date and accurate (sorry about the ?? in various places).

ST	#	Name	Day/Time	City
AL	19	WJTC	??	Biloxi
	42	WEIQ	Sat 9pm	Mobile
AZ	8	KAET	Sun noon	Tempe
CA	9	KIXE	Sat 10pm	Redding
	28	KCET	Sat 9:30am	Los Angeles
	6	KVIE	??	Sacramento
	54	KTEH	??	San Jose
CO	6	KRMA	Sun 10am	Denver
CT	24	WEDH	Sat 6pm	Hartford
DC	26	WETA	Sun 11am	Washington
DE	64	WDPB	Sat 3pm	Seaford
	64	WDPB	Fri Midnite	Seaford
FL	2	WPBT	??	Vero Beach
	24	WMFE	Sat 10pm	Orlando
IL	12	WILL	M-F 10pm	Urbana
	11	WTTW	Sun 11pm	Chicago
IN	20	WFYI	Sat 10:30pm	Indianapolis
	39	WFWA	Sat 11:30pm	Fort Wayne
LA	24	KLTS	Sat 10:30pm	Shreveport
MA	2	WGBH	Sun 11pm	Boston
	57	WGBY	Sat 6pm	Springfield
MD	22	WMPT	Sat 11pm	Annapolis
	67	WMPB	Sat 11pm	Baltimore
	28	WCPB	Sat 11pm	Salisbury
MI	56	WKBD	Sun 11pm	??
	56	WTVS	Sun 11pm	Ann Arbor
	??	WGVU	??	??
MI	52	WGVK	Sun 4pm	Grand Rapids
	35	WGVC	Sat 4pm	Allendale
MN	2	KTCA	Sat 8am	Minneapolis
	15	KSMQ	Sat 8pm	??
	2	KTCA	Sat 10pm	Minneapolis
MO	19	KCPT	Fri 10:30pm	Kansas City
	9	KETC	Sun 10pm	St Louis
MS	45	W45AA	Sat 4pm	Columbia
	14	WMAW	Sat 4pm	Meridian
	17	WMAU	Sat 4pm	Bude
	23	WMAO	Sat 4pm	Greenwood
	19	WMAH	Sat 4pm	Biloxi
	12	WMAE	Sat 4pm	Booneville
	2	WMAB	Sat 4pm	Mis State
	29	WMAA	Sat 4pm	Jackson
	18	WMAV	Sat 4pm	Oxford Univ
NC	58	WUNG	M-F 11pm	Charlotte
	19	WUNM	M-F 11pm	Jacksonville
	26	WUNL	M-F 11pm	WinstonSalem
	39	WUNJ	M-F 11pm	Willmington
	33	WUNF	M-F 11pm	Asheville
	2	WUND	M-F 11pm	Columbia
	4	WUNC	M-F 11pm	Chapel Hill
	25	WUNK	M-F 11pm	Greenville
	36	WUNP	M-F 11pm	RoanokeRpds
	17	WUNE	M-F 11pm	Linville
NH	11	WEHN	Sat 5pm	NH
NJ	50	WNJM	Sat 9pm	Montclair
	58	WNJB	Sat 9pm	NJ
	52	WNJT	Sat 9pm	Wildwood
	23	WNJS	Sat 9pm	Camden
NY	17	WNED	Sat 4:30pm	Buffalo
	21	WLIW	Sat 5:30	Long Island
	39	??	M-F 11:30pm	??
OH	48	WCET	Sat 10:30pm	Cincinati
	34	WOSU	Sat 10pm	Columbus
PA	23	WITF	??	Hershy
	12	WHYY	Fri Midnite	Philadelphia
	12	WHYY	Sat 3pm	Philadelphia
TN	8	WDCN	Tue 7pm	Nashville
TX	13	KERA	Sat 10pm	WichitaFalls
	??	KEDT	Sat 10pm	CorpusChrsti
	13	KCOS	Sat 9pm	El Paso
WA	??	KTPS	Sat 8pm	Tacoma
WI	36	WLEF	Sun Noon	Park Falls
	28	WHWC	Sun Noon	Menominee
	20	WHRM	Sun Noon	Wausau
	31	WHLA	Sun Noon	La Crosse
	21	WHA	Sat 10am	Madison
	38	WPNE	Sun Noon	Green Bay
??	56	WUCM	Sat 10pm	??
??	23	WKAR	Sun 2pm	??
??	19	WFUM	Sun 11pm	??
??	31	WNYC	Fri 9pm	??

GERMAN AMIGA COMAL 2.0 SUPPLIED PACKAGES SUMMARY

(Preliminary Specifications Subject To Change)

WINDOWS

AllocWindow*(X,Y,Width,Height,Name$)*
CloseWindow*(Window)*
CloseWindows
FreeWindow*(Window)*
FreeWindows
MoveWindow*(Window,dX,dY)*
OpenWindow*(NewWindow)*
WindowScreen*(Screen)*
WindowToBack*(Window)*
WindowToFront*(Window)*

SCREENS

AllocScreen*(Mode640,Interlace,Depth,Font,Name$)*
CloseScreen*(Screen)*
CloseScreens
CloseWorkBench
FreeScreen*(Screen)*
FreeScreens
MoveScreen*(Screen,dY)*
OpenScreen*(NewScreen)*
OpenWorkBench
ScreenToBack*(Screen)*
ScreenToFront*(Screen)*
WorkBenchToBack
WorkBenchToFront

SYSTEM

AllocMemory*(ByteSize,Requirements)*
FreeMemory*(Address)*
FreeAll
Bin$*(Number)*
Hex$*(Number)*
Poke_B*(Address,ByteValue)*
Poke_W*(Address,WordValue)*
Poke_L*(Address,LongValue)*
Poke_S*(Address,String$)*
Peek_B*(Address)*
Peek_W*(Address)*
Peek_L*(Address)*
Peek_S$*(Address)*
Sto_Address*(Address)*
Sto_B*(ByteValue)*
Sto_W*(WordValue)*
Sto_L*(LongValue)*

BORDERS

AllocBorder*(N)*
Border*(Bhandle,N,X,Y)*
BorderColor*(Bhandle,Color)*
DrawBorder*(Window,Bhandle,X,Y)*
FreeBorder*(Bhandle)*
FreeBorders

IMAGES

AllocImage*(Width,Height,Depth)*
DrawImage*(Window,Image,X,Y)*
FreeImage*(Image)*
FreeImages

TEXTS

AllocText*(X,Y,Font,Text$)*
DrawText*(Window,Text,X,Y)*
FreeText*(Text)*
FreeTexts

NARRATOR

Pronounce*(Text$)*
**Translate$***(Text$)*

SPEECH

Say*(Text$)*

PCGRAPHICS

GraphicScreen*(Mode)*
TextScreen
ViewPort*(Xmin,Xmax,Ymin,Ymax)*
Window*(Xmin,Xmax,Ymin,Ymax)*
Clear
PenColor*(Color)*
MoveTo*(X,Y)*
Move*(X,Y)*
DrawTo*(X,Y)*
Draw*(X,Y)*
Plot*(X,Y)*
PlotText*(Text$)*
PlotText*(Text$,X,Y)*
ReadPixel*(X,Y)*
Width
Height
Depth
Circle*(Radius,X,Y)*
Fill*(X,Y)*

TURTLE

bk*(x) // backward*
cs *// clear screen*
fd*(x) // forward*
home
ht *// hideturtle*
lt*(v) // left*
pd *// pendown*
pu *// penup*
rt*(v) // right*
seth*(v) // setheading*
st *// showturtle*